Coursework Coversheet

School of Geography. FACULTY OF ENVIRONMENT

GEOG5403 Creative Coding for Urban Problems

UNIVERSITY OF LEEDS

Student ID	201788236/201787698/201790263/201776257/201793166/201788996	Mark Less deduction (state reason) Final Mark	
Group Number	Group 5		
Assignment title	Child labour in global supply chain		
Marker			

Justification of mark (using specific text from criteria):

To improve your work for next time:
1.
2.
3.

Additional comments:

Group 5

Child labour in global supply chain

Alexander Thornton, Shenghang Yuan, Subaru Shimizu, Yuxi Liu, Zhaojing Wang,Zhihao Zhang

Background

Nearly 1 in 10 children are subjected to child labour worldwide,
With some forced into hazardous work through trafficking.

Why do families choose children to be employed

- Economic hardship
- Migrant and refugee children

Why companies use child labour

- High Demand, Low Skill
- Lower price
- Less Oversight

Background

Nearly 1 in 10 children are subjected to child labour worldwide,
With some forced into hazardous work through trafficking.

Why child labour data is not available/difficult to obtain

- Informal sectors
- Hidden or illegal contexts
- Reluctance to report

Consequence

- Extreme bodily and mental harm
- Compound social inequality and discrimination

Approach to the challenge and Used Data

1. **Challenge:** It is difficult to understand the current situation of child labour due to the data availability as companies tend to hide negative information.
2. **Objective:** To understand where, in which commodity production and by which company the child labour is executed and explore ways to solve these problems.
3. **Approach:** Use the news article provided by HACE, and then extract articles that contain our keywords, extract information using ChatGPT API, and finally visualize the results.
4. **Used dataset:** Text files of news articles by HACE, we mainly selected dataset from different countries, companies and commodities to analyze and see the relationship between them.

Flow of Analysis

1. **Identification of keywords:** Keywords related to child labour and commodities were identified. These keywords were used in following steps to filter and extract relevant news articles.
2. **Reading articles:** These articles were obtained from some data source, for example a news website.
3. **Extracting articles that contain our keywords:** We identified each text file to figure out whether it contains keywords (child labour). If so, this file was extracted for analysis.
4. **Extracting key information using ChatGPT API:** We used ChatGPT to process and analyse text files to obtain information, including countries, commodities and companies involved.
5. **Visualisation (including minor data cleaning):** The extracted information was used for visualisation. It includes the use of interactive maps to show the distribution of countries, and the display of information about commodities and companies.

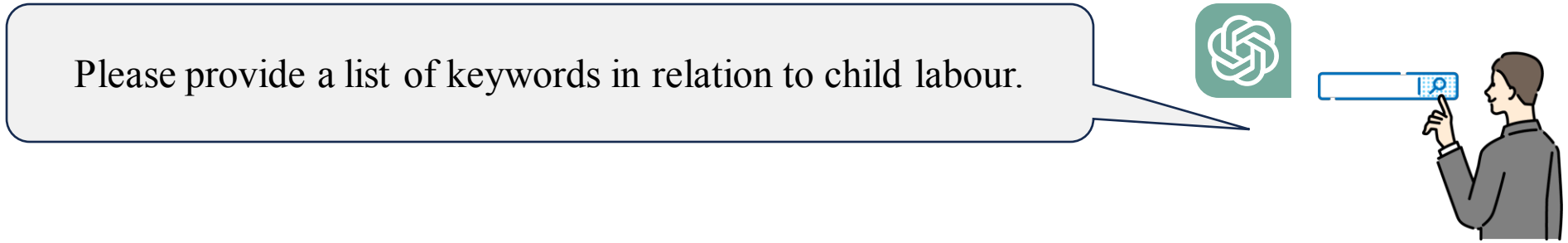
Methods

Challenge 1: Massive dataset that contains news articles irrelevant to child labour

- The provided dataset has 4,561 news article files, which is too massive, and contains ones irrelevant to child labour.

Solution 1: Extracted news articles that contain child labour-related keywords

- Prepared a list of 61 child-labour-related keywords with the assistance of ChatGPT.
  - *Examples: Child labour, Child labor, Kid worker, Child exploitation, Child trafficking, Child slavery, Child bondage, Juvenile exploitation, etc.*
- Extracted news articles that contain child labour-related keywords.



Methods

Challenge 2: Difficulty in extracting key information from news articles.

Example:

child labor taints production of batteries for electric carmakers, amnesty says news today's news us politics world tech science weather opinion originals the 360 skullduggery podcast conspiracyland finance my portfolio yahoo finance plus news screeners markets videos watchlists personal finance crypto industries sports fantasy daily fantasy nfl nba mlb nhl ncaaf ncaab more ncaaw mma sportsbook soccer tennis nascar golf boxing cycling wnba usfl indycar horse racing olympics gamechannel rivals podcasts videos rss team apparel and gear shop breakingt shirts newsfeed entertainment celebrity tv movies music life health style and beauty parenting horoscopes wel-being food travel terms privacy privacy dashboard feedback © 2023 all rights reserved. about our ads advertising careers

yahoo news search query news finance sports more news today's news us politics world covid-19 climate change health science originals life health parenting style and beauty horoscopes shopping entertainment role recall anniversary party game changers the it list under the covers the neverweres are the kids alright? how to watch interviews videos shopping finance my portfolio watchlists markets news videos yahoo finance plus screeners personal finance crypto industries sportsfantasy nfl nba mlb nhl college football college basketball soccer mma yahoo sports am editions us englishus españolaustralia englishcanada englishcanada françaisfrance françaisgermany deutschhong kong 繁體中文malaysia englishsingapore englishtaiwan 繁體中文uk english mail sign in advertisementclose this contentreuterschild labor taints production of batteries for electric carmakers, amnesty saysby lin taylorseptember 30, 2016 at 3:38 amlink copiedread full articleby lin taylor london (thomson reuters foundation) - leading electric carmakers may be unwittingly using child labor to produce batteries for vehicles that have grownin popularity for using clean energy aimed at limiting global warming, amnesty international said onfriday. the human rights watchdog said cobalt used in lithium ion batteries for electric vehicles, phones and laptops could come from mines in democratic republic of congo (drc) that use child labor. it accused carmakers including gm, renault-nissan, fiat chrysler, volkswagen, daimler and tesla of failing to map the supply of cobalt from mines in congo to smelters and on to battery-makers. as a result, electric cars sold across the globe could contain traces of the metal produced each year by informal congolese mines without companies knowing, the group said. "we have found that there is significant risk that cobalt mined by children could be entering their supply chains," said markdummett, human rights researcher at amnesty. due to the growing demand for lithium ion batteries in electric cars and consumer goods like mobile phones, dummett said carmakers had an obligation to trace their supply chain to ensure children were not used to mine cobalt. "frankly companies owe it to their consumer to be transparent about their supplies and to map out their supply chains so that they know where it's coming from," he said in an interview with the thomson reuters foundation. more than half of the world's cobalt comes from drc, amnesty said, and 20 percent of the mineral was mined by hand. millions of congolese work in informal mining, with rudimentary tools and usually without legal

This article mentions the possibility that cobalt mined by children in DRC is supplied to EV manufacturers.

Key information

- **Country**: Democratic Republic of Congo (DRC)
- **Company**: GM, Renault-Nissan, Fiat Chrysler, Volkswagen, Daimler, Tesla
- **Commodity**: Cobalt

Source of figure: CNN

Methods

Challenge 2: Difficulty in extracting key information from news articles.

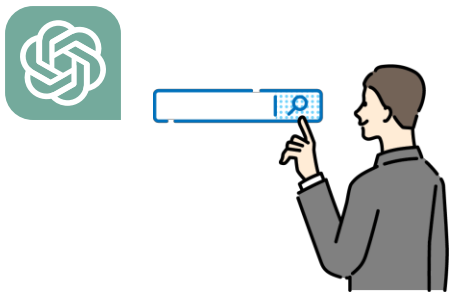
Solution 2: Extracted key information using ChatGPT API.

- Utilised ChatGPT API to extract key information from news articles.
- Provided ChatGPT with a list of 62 relevant commodities to improve returned answers.

Can you find the **companies** that are mentioned in relation to child labour exploitation?
Here is an example of your message: 'Microsoft, Alphabet'.

Can you find the **countries** that are mentioned in relation to child labour exploitation?
Here is an example of your message: 'United States, Indonesia'.

Can you find the **commodities** that are mentioned in relation to child labour exploitation?
Please provide a list of commodity names, using the names in the commodity list: '{commodity_labour}'.
Here is an example of your message: 'Jewelry, Apparel'.



Relevant commodities: Gold, Cobalt, Textiles, Apparel, Footwear, Jewellery, Batteries, etc.

Result Description

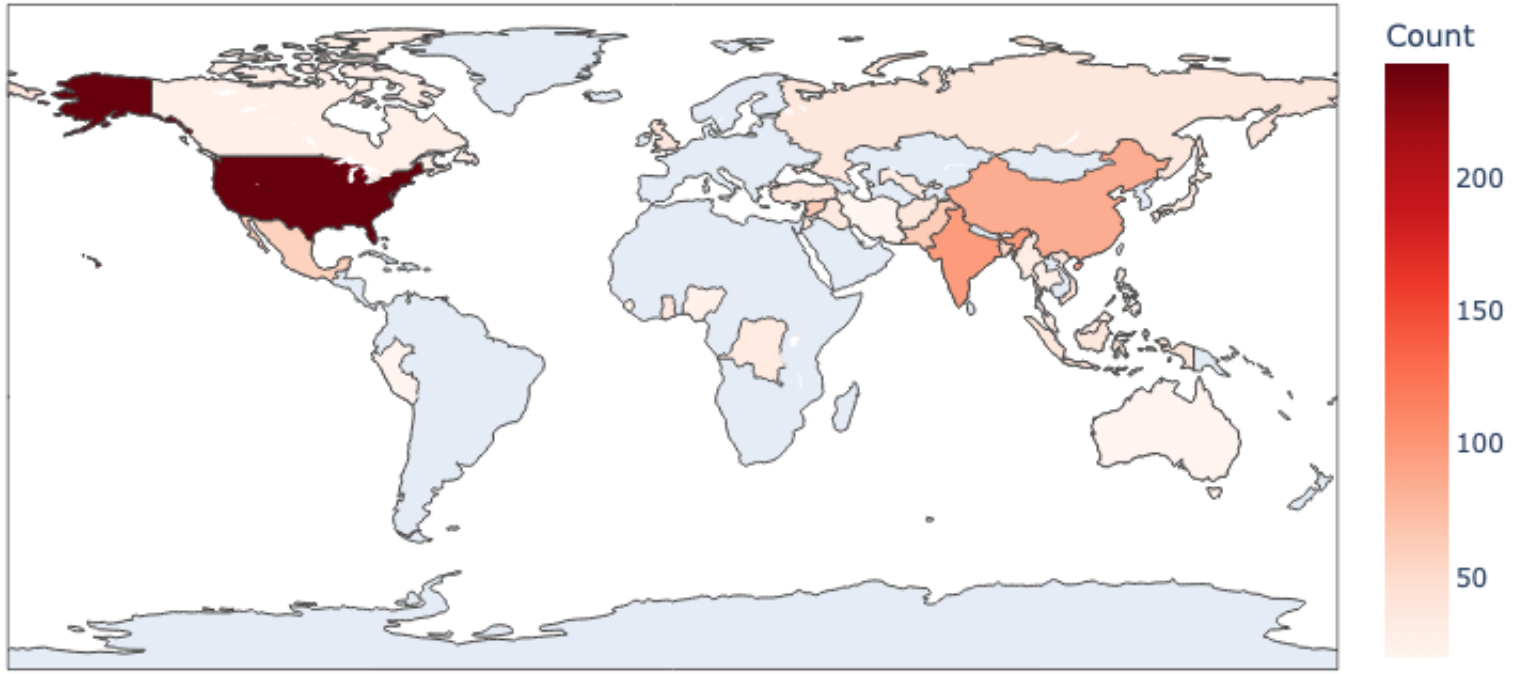
Summarised returned answers from ChatGPT.

- Company
- Commodity
- Country

Company	Commodity	Country
None.	None	Brazil
Apple, Tesla	Cobalt	Democratic Republic of Congo
Apple Inc, Sony Corp, Volvo	Cobalt	Democratic Republic of Congo
None	Cotton	None
None	'Cobalt'	Canada, Democratic Republic of Congo
None	Child Labor, Clothing, Footwear, Toys	United States, Germany, Japan, Nicaragua, Mexico, United Arab Emirates, Kuwait, Oman, Georgia, Florida, Alabama, Alaska, Tennessee, Indiana, Oklahoma, Louisiana
None	Gravel, Construction materials, Stone, Fish, Fish processing	Myanmar
None	None	United States
None	Cotton	Turkmenistan
None	Toys, Handicrafts, Textiles, Apparel, Carpentry, Garments, Pharmaceuticals.	Bolivia
None	None	Brazil, United States
None	Fish, Coffee, Sugarcane, Cotton, Tobacco, Alcohol, Leather, Cattle	China, India, Bangladesh, Brazil, Indonesia, Philippines, Vietnam, Thailand, Egypt, Cambodia, Pakistan, Malaysia, Myanmar, Turkey, Mexico, Nigeria, Ghana, Kenya, El Salvador, Guatemala, Honduras, Nicaragua, Panama, Colombia, Costa Rica, Dominican Republic, Haiti, Jamaica
None	None	None.
cez, korea electric power (kepco), norges bank investment management (nbim)	Coal, Textiles	Turkey
None	Agricultural produce, Textiles, Apparel, Construction materials	Vietnam
None	None	Jordan, United States, Syria, Iraq, Yemen, Libya, Pakistan
None	None	Chile, Yemen, Syria, Iraq, Jordan, the Netherlands, Monaco, East Timor, Australia, India
None	Toys	China

Visualisations

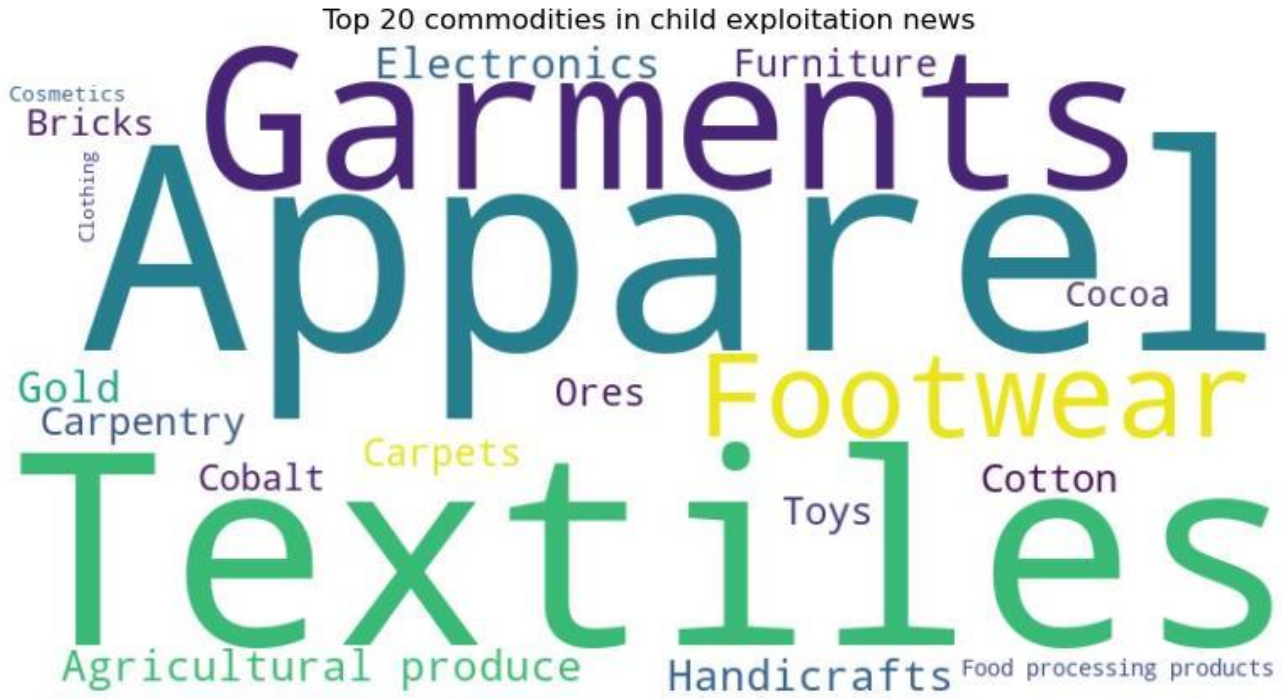
Frequencies of child exploitation news by country (over 20 counts only)



Country	Count
United States	243
India	97
China	86
Syria	64
Pakistan	63
Mexico	58
Bangladesh	54
Afghanistan	39
Ghana	39
Russia	39
Indonesia	38
United Kingdom	38
Vietnam	38
Turkey	37
Iraq	36
Uzbekistan	35
Democratic Republic of Congo	33
Myanmar	30
Jordan	30
Malaysia	29

Visualisations

Commodities:



Commodity	Count
Apparel	115
Textiles	101
Garments	61
Footwear	57
Agricultural produce	45
Electronics	44
Handicrafts	43
Gold	43
Cotton	37
Toys	37
Carpentry	36
Ores	32
Carpets	31
Furniture	30
Cobalt	28
Bricks	27
Cocoa	25
Food processing products	25
Cosmetics	22
Clothing	18

Visualisations

Companies:

Company	Count
NESTLE	20
WALMART	15
H&M	14
HERSHEY	11
GAP	10
SAMSUNG	9
APPLE	9
MARS	9
NIKE	8
TESCO	7
CARGILL	7
ADIDAS	6
SAVE THE CHILDREN	6
PRIMARK	6
FERRERO	6
INDITEX	6
FOXCONN	5
PEPSICO	5
THAI UNION	5
TESLA	5
MONDELEZ	5
DE BEERS	5

Findings from Analysis

Case 1: United States

Company and Commodity	• **Driscoll's and Walmart:** Agricultural products such as berries and vegetables • **Coca-Cola and other beverage manufacturers:** the production of sugar and other agricultural products, especially cocoa, fruits and vegetables.
Insights from articles	• In agriculture and textile manufacturing, companies may exploit child labour to perform **low-skilled and labour-intensive work** in order to reduce costs. In agriculture, in particular, child labour is easier to overlook because production sites are scattered and difficult to regulate. • Farms in the United States rely on **the immigrant labour force (Mexican workers)** and exploit these people.

Case 2: China

Company and Commodity	• **Electronics** and **clothing/textile** are two major commodities associated with child labour, involving several internationally renowned brands such as **Foxconn, Apple, Gap and H&M**.

Solutions and Policies


https://www.google.com

Strengthening regulation and enforcement

- Strengthen regulations to ensure that enterprises, including their suppliers and subcontractors in developing countries, comply with international standard labour requirements.
- Strengthen enforcement to punish businesses that violate labour laws.
- Encourage companies to inspect their supply chains to ensure that child labour is not involved.

Improve education and economic conditions

- Provide more educational resources such as free textbooks and uniforms to ensure more children have access to good education.
- Provide financial support and job opportunities to those who have children.

Create social co-operation

- Work together to address child labour through cooperation between government, business and the community. Establish a social support system to provide necessary assistance to children exploited by child labour.

Future work

Preparing readable dataset

- Cleaning/reformatting the dataset by removing unnecessary texts such as headers/footers will improve readability.

Improving prompts given to ChatGPT for better accuracy

- Providing ChatGPT with a distinct prompt, including background knowledge and specific areas of concern, to return our expected responses.

Developing natural language processing models for sustainability and accountability

- Since relying on ChatGPT API can be costly, advanced natural language processing models (e.g., entity recognition systems) that improve accuracy and accountability will be recommended.

Improving output data cleaning and validation process

- Responses from ChatGPT require manual data cleaning, such as unifying names (e.g. Apple, Apple Inc.). Giving a list of companies or commodities will improve the output data.
- Due to the complex texts, the output needs validation to ensure accuracy.

References

Anon n.d. *Child labor versus educational attainment Some evidence from Latin America Original*.

Edmonds, E.V. and Schady, N. 2012. Poverty alleviation and child labor. *American Economic Journal. Economic Policy*. **4**(4), pp.100–124.

Humphries, J. 2013. Childhood and child labour in the British industrial revolution1: CHILDHOOD AND CHILD LABOUR. *The economic history review*. **66**(2), pp.395–418.

Nieuwenhuys, O. 1996. The paradox of child labor and anthropology. *Annual review of anthropology*. **25**(1), pp.237–251.

Ray, R. 2000. Child labor, child schooling, and their interaction with adult labor: Empirical evidence for Peru and Pakistan. *The World Bank economic review*. **14**(2), pp.347–367.

Zapata, D., Contreras, D. and Kruger, D. 2011. Child labor and schooling in Bolivia: Who's falling behind? The roles of domestic work, gender, and ethnicity. *World development*. **39**(4), pp.588–599.

Contribution

Coding: All group numbers

Background and Approach: Shenghang Yuan

Data description and Methods: Alexander Thornton, Zhihao Zhang

Visualisation and Future work: Subaru Shimizu

Slides creation and design: Zhaojing Wang, Yuxi Liu

Speech: Alexander Thornton, Subaru Shimizu, Zhaojing Wang

Thanks

In [2]:
```
import zipfile
import os

def extract_zip(zip_file, extract_to):
    with zipfile.ZipFile(zip_file, 'r') as zip_ref:
        zip_ref.extractall(extract_to)

# # Example usage
# zip_file = 'OneDrive_2024-03-19.zip'  # Replace 'example.zip' with the name of your zip file
# extract_to = r'C:\Users\23510\Creative Coding for Urban Problems\Week 4'  # Replace 'extracted_files' with the directory where you want to extract the files

# # Create the directory if it doesn't exist
# os.makedirs(extract_to, exist_ok=True)

# # Extract the zip file
# extract_zip(zip_file, extract_to)

```

In [24]:
```
import os

import pandas as pd
import numpy as np
import seaborn as sns
palette = 'bwr'
sns.set_palette(palette)

import matplotlib.pyplot as plt
from matplotlib.pyplot import figure
pd.options.mode.chained_assignment = None

#import datetime
from datetime import datetime

import plotly.express as px
import plotly.graph_objects as go
import plotly.io as pio
import plotly.offline as pyo

%matplotlib inline

from pysal.lib import weights
from pysal.lib import io

import contextily as cx
import geopandas as gpd
from shapely.geometry import Point
from shapely import wkt

import folium
from folium import plugins

import spacy
import openai
import requests

from wordcloud import WordCloud
import matplotlib.pyplot as plt

pd.set_option('display.max_columns', 200)
pd.set_option('display.max_rows', 200)
```

In [14]:
```
# Assign directory

directory = r'/Users/subaru/Library/CloudStorage/OneDrive-UniversityofLeeds/Courses/02_Sem2/5403_Creative Coding/Week4/Alex/dataset'

# get all files under directory
file_list = os.listdir(directory)

# change the extention to .txt
for file_name in file_list:
    # file's full path
    file_path = os.path.join(directory, file_name)
    # change the exrtention
    new_file_path = os.path.splitext(file_path)[0] + '.txt'
    try:
        # change the file name
        os.rename(file_path, new_file_path)
        print(f"Renamed {file_path} to {new_file_path}")
    except Exception as e:
        print(f"Error renaming file {file_path}: {e}")
```

```
e/OneDrive-UniversityofLeeds/Courses/02_Sem2/5403_Creative Coding/Week4/Alex/dataset/20170123191500-894.txt
Renamed /Users/subaru/Library/CloudStorage/OneDrive-UniversityofLeeds/Courses/02_Sem2/5403_Creative Coding/Week4/Alex/dataset/20160604200000-T2680.txt to /Users/subaru/Library/CloudStor
age/OneDrive-UniversityofLeeds/Courses/02_Sem2/5403_Creative Coding/Week4/Alex/dataset/20160604200000-T2680.txt
Renamed /Users/subaru/Library/CloudStorage/OneDrive-UniversityofLeeds/Courses/02_Sem2/5403_Creative Coding/Week4/Alex/dataset/20161008091500-484.txt to /Users/subaru/Library/CloudStorag
e/OneDrive-UniversityofLeeds/Courses/02_Sem2/5403_Creative Coding/Week4/Alex/dataset/20161008091500-484.txt
Renamed /Users/subaru/Library/CloudStorage/OneDrive-UniversityofLeeds/Courses/02_Sem2/5403_Creative Coding/Week4/Alex/dataset/20161118131500-1982.txt to /Users/subaru/Library/CloudStora
ge/OneDrive-UniversityofLeeds/Courses/02_Sem2/5403_Creative Coding/Week4/Alex/dataset/20161118131500-1982.txt
Renamed /Users/subaru/Library/CloudStorage/OneDrive-UniversityofLeeds/Courses/02_Sem2/5403_Creative Coding/Week4/Alex/dataset/20151201211500-2747.txt to /Users/subaru/Library/CloudStora
ge/OneDrive-UniversityofLeeds/Courses/02_Sem2/5403_Creative Coding/Week4/Alex/dataset/20151201211500-2747.txt
Renamed /Users/subaru/Library/CloudStorage/OneDrive-UniversityofLeeds/Courses/02_Sem2/5403_Creative Coding/Week4/Alex/dataset/20161125101500-225.txt to /Users/subaru/Library/CloudStorag
e/OneDrive-UniversityofLeeds/Courses/02_Sem2/5403_Creative Coding/Week4/Alex/dataset/20161125101500-225.txt
Renamed /Users/subaru/Library/CloudStorage/OneDrive-UniversityofLeeds/Courses/02_Sem2/5403_Creative Coding/Week4/Alex/dataset/20170615204500-612.txt to /Users/subaru/Library/CloudStorag
e/OneDrive-UniversityofLeeds/Courses/02_Sem2/5403_Creative Coding/Week4/Alex/dataset/20170615204500-612.txt
Renamed /Users/subaru/Library/CloudStorage/OneDrive-UniversityofLeeds/Courses/02_Sem2/5403_Creative Coding/Week4/Alex/dataset/20161124071500-T703.txt to /Users/subaru/Library/CloudStora
ge/OneDrive-UniversityofLeeds/Courses/02_Sem2/5403_Creative Coding/Week4/Alex/dataset/20161124071500-T703.txt
Renamed /Users/subaru/Library/CloudStorage/OneDrive-UniversityofLeeds/Courses/02_Sem2/5403_Creative Coding/Week4/Alex/dataset/20160609040000-247.txt to /Users/subaru/Library/CloudStorag
e/OneDrive-UniversityofLeeds/Courses/02_Sem2/5403_Creative Coding/Week4/Alex/dataset/20160609040000-247.txt
Renamed /Users/subaru/Library/CloudStorage/OneDrive-UniversityofLeeds/Courses/02_Sem2/5403_Creative Coding/Week4/Alex/dataset/20160612194500-T922.txt to /Users/subaru/Library/CloudStora
ge/OneDrive-UniversityofLeeds/Courses/02_Sem2/5403_Creative Coding/Week4/Alex/dataset/20160612194500-T922.txt
Renamed /Users/subaru/Library/CloudStorage/OneDrive-UniversityofLeeds/Courses/02_Sem2/5403_Creative Coding/Week4/Alex/dataset/20170503060000-T556.txt to /Users/subaru/Library/CloudStora
```

Step 1: Preparation

1-1: Read Keywords of Child Labour and Commodities

In [15]:
```
# Child labour related keywords
keywords_excel = pd.read_excel('key_words.xlsx')
keywords_excel.head()
```

Out[15]:

	Relevent to Child labour	Relevent to commodies	Fortune 500
0	Child labour	Gold	Walmart Inc.
1	child labor	Cobolt	Amazon.com Inc.
2	kid worker	Textiles	Apple Inc.
3	Child Exploitation	Apparel	Exxon Mobil Corporation
4	Underage Labour	Footwear	Berkshire Hathaway Inc.

In [16]:
```
keyword_labour = []
for words in keywords_excel['Relevent to Child labour']:
    keyword_labour.append(words)

keyword_labour = [keyword for keyword in keyword_labour if pd.notnull(keyword)]
keywords = keyword_labour
keywords
```

Out[16]:
```
['Child labour',
 'child labor',
 'kid worker',
 'Child Exploitation',
 'Underage Labour ',
 'Juvenile Labour',
 'Child Work',
 'Child Employment ',
 'Minor Labour',
 'Youth Labour',
 'Adolescent Labour',
 'Child Servitude',
 'Child Trafficking',
 'Child Servitude',
 'Underage Employment',
 'Child Exploitation',
 'Minor Workforce',
 'Youth Exploitation',
 'Child Trafficking',
 'Juvenile Exploitation',
 'Child Bondage',
 'Child Forced Labour',
 'Child Slavery',
 'Child Indentured Labour',
 'Child Sweatshop Labour',
 'Child Exploitative Labour',
 'Child Labor Trafficking',
 'Child Commercial Exploitation',
 'Child Abuse of Labour',
 'Child Employment Exploitation',
 'Child Labor Exploitation',
 'Child Labor Abuse',
 'Child Labor Trafficking',
 'Child Work Exploitation',
 'Child Work Abuse',
 'Child Labor Violation',
 'Child Workforce Exploitation',
 'Child Labor Malpractice',
 'Child Labor Infringement',
 'Child Workforce Injustice',
 'Child Labor Iniquity',
 'Child Labor Misuse',
 'Child Labor Maltreatment',
 'Child Abuse of Labour',
 'Child Sexual Exploitation',
 'Child Labor Exploitation',
 'Child Labor Abuse',
 'Child Pornography',
 'Child Labor Trafficking',
 'Child Labor Violation',
 'Child Labor Infringement',
 'Child Prostitution',
 'Child Marriage',
 'Child Sex Trafficking',
 'Child Labor Misuse',
 'Child Labor Maltreatment',
 'Child Labor Iniquity',
 'Child Commercial Exploitation',
 'Child Soldiering',
 'Child Domestic Servitude',
 'Child Labor in Agriculture']
```

In [17]:
```
# Commodity related keywords
commodity_labour = []
for words in keywords_excel['Relevent to commodies']:
    commodity_labour.append(words)

commodity_labour = [keyword for keyword in commodity_labour if pd.notnull(keyword)]
commodity_labour

```

Out[17]:
```
['Gold',
 'Cobolt\xa0',
 'Textiles',
 'Apparel',
 'Footwear',
 'Toys',
 'Handicrafts',
 'Electronics',
 'Cosmetics',
 'Furniture',
 'Food processing products',
 'Ores',
 'Bricks',
 'Recycled materials',
 'Paper products',
 'Utensils',
 'Medical supplies',
 'Packaging materials',
 'Agricultural produce',
 'Metal products',
 'Building materials',
 'Leather goods',
 'Carpets',
 'Rugs',
 'Carpentry',
 'Garments',
 'Watches',
 'Jewelry',
 'Glassware',
 'Ceramics',
 'Pottery',
 'Tiles',
 'Plastics',
 'Paintings',
 'Sculptures',
 'Musical instruments',
 'Watches',
 'Eyewear',
 'Cosmetics',
 'Pharmaceuticals',
 'Automotive parts',
 'Sporting goods',
 'Fireworks',
 'Matches',
 'Cigarettes',
 'Handbags',
 'Wallets',
 'Stationery',
 'Carpentry tools',
 'Construction materials',
 'Cleaning products',
 'Agricultural chemicals',
 'Textile dyes',
 'Glues and adhesives',
 'Paints and coatings',
 'Batteries',
 'Electrical components',
 'Plastics and rubber products',
 'Carpets and rugs',
 'Furniture upholstery',
 'Disposable cutlery and dishes',
 'Fishing gear']
```

1-2: Read Text (News Article) Files

In [18]:
```
def extract_text_file(file_path, encoding='utf-8'):
    with open(file_path, 'r', encoding=encoding) as file:
        text = file.read()
    return text

# Example usage
file_path = r'/Users/subaru/Library/CloudStorage/OneDrive-UniversityofLeeds/Courses/02_Sem2/5403_Creative Coding/Week4/Alex/dataset/20150310204500-2046.txt'
# Replace with the path to your text file
text_content = extract_text_file(file_path)
print(text_content)

```

```
new tax shocker for middle class to cater for the elderly, orphans - the standard

×

 the standard group plc is a
multi-media organization with investments in media platforms spanning newspaper
print operations, television, radio broadcasting, digital and online services. the
standard group is recognized as a leading multi-media house in kenya with a key
influence in matters of national and international interest.

standard group plc hq office,
the standard group center,mombasa road.
p.o box 30080-00100,nairobi, kenya.
telephone number: 0203222111, 0719012111
 email: [email  protected]

news & current affairs

digital news
```

1-3: Find Text files related to Child labour

In [19]:
```
## Find text files that contains the keywords.
def find_child_labor_references_in_folder(folder_path, keywords, max_files=10000, encoding='utf-8'):
    child_labor_files = []

    # Iterate over files in the folder
    file_count = 0
    for filename in os.listdir(folder_path):
        if file_count >= max_files:
            break

        if filename.endswith('.txt'):
            file_path = os.path.join(folder_path, filename)
            with open(file_path, 'r', encoding=encoding, errors='ignore') as file:
                text = file.read().lower()  # Convert text to lowercase for case-insensitive matching

            # Check for keywords in the text
            for keyword in keywords:
                if keyword in text:
                    child_labor_files.append(filename)
                    break  # Once any keyword is found, no need to check further for this file

            file_count += 1

    return child_labor_files

# Example usage
folder_path = r'/Users/subaru/Library/CloudStorage/OneDrive-UniversityofLeeds/Courses/02_Sem2/5403_Creative Coding/Week4/Alex/dataset'  # Replace with the path to your folder
child_labor_files = find_child_labor_references_in_folder(folder_path, keywords)

if child_labor_files:
    print("Files mentioning child labor keywords:", child_labor_files)
else:
    print("No files mentioning child labor keywords found.")

```

```
30074500-1584.txt', '20160825110000-2287.txt', '20180305224500-1251.txt', '20170612234500-1498.txt', '20170320201500-338.txt', '20170405181500-522.txt', '20160119223000-781.txt', '20151106171500-1092.txt', '20170211000000-1450.txt', '20180329100000-429.txt', '20161103214500-2066.txt', '20150713191500-1901.txt', '20160901183000-191.txt', '20170316043000-167.txt', '20180411233000-2775.txt', '20180425054500-1739.txt', '20160402161500-301.txt', '20161021190000-633.txt', '20160419201500-448.txt', '20151022224500-1531.txt', '20160614101500-1857.txt', '20160704191500-2891.txt', '20160706134500-2513.txt', '20151008004500-2494.txt', '20150905001500-92.txt', '20150829010000-542.txt', '20160621171500-1903.txt', '20180407230000-495.txt', '20180415034500-82.txt', '20170320111500-2508.txt', '20160114024500-168.txt', '20170428010000-2402.txt', '20170616051500-1786.txt', '20180418161500-652.txt', '20161129074500-718.txt', '20180313174500-278.txt', '20160315140000-2817.txt', '20160527061500-1056.txt', '20170802233000-2017.txt', '20170817060000-1186.txt', '20151205174500-1945.txt', '20180421163000-1592.txt', '20160305014500-121.txt', '20170402200000-685.txt', '20170207084500-994.txt', '20160212160000-2058.txt', '20180302184500-2700.txt', '20170320071500-2049.txt', '20170331131500-1255.txt', '20170623014500-93.txt', '20151113161500-674.txt', '20170728133000-1361.txt', '20150424151500-1097.txt', '20161102043000-2335.txt', '20170422044500-636.txt', '20160202223000-550.txt', '20160710110000-1155.txt', '20180422051500-794.txt', '20170330180000-666.txt', '20160115080000-148.txt', '20150703054500-819.txt', '20170315120000-1536.txt', '20180314154500-2922.txt', '20160812054500-2455.txt', '20161224174500-507.txt', '20160920061500-2425.txt', '20170621163000-3.txt', '20170427111500-2058.txt', '20160205230000-284.txt', '20180418133000-1866.txt', '20170120173000-1073.txt', '20180328200000-2221.txt', '20170316160000-1788.txt', '20150501161500-2598.txt', '20170214033000-2022.txt', '20170315210000-177.txt', '20170322180000-1462.txt', '20170620003000-179.txt', '20160105114500-1354.txt', '20160422220000-809.txt', '20160613163000-1090.txt', '20170214170000-566.txt', '20170522223000-2506.txt', '20170722204500-1934.txt', '20170421134500-361.txt', '20180413134500-1401.txt', '20160503123000-427.txt', '20180403190000-1525.txt', '20150717211500-2346.txt', '20170614174500-804.txt', '20180404171500-654.txt', '20160808153000-2848.txt', '20160906164500-901.txt', '20160122233000-2899.txt', '20160428173000-1350.txt', '20180309040000-234.txt', '20170615203000-286.txt', '20161213070000-1956.txt', '20170323000000-2337.txt', '20170311063000-1680.txt', '20160725103000-2219.txt', '20170521000000-274.txt', '20150501003000-1733.txt', '20180416113000-2586.txt', '20160429044500-316.txt', '20180419201500-2398.txt', '20151218170000-1875.txt', '20170306111500-2058.txt', '20180403190000-986.txt', '20170413204500-2870.txt', '20160630131500-558.txt', '20160409034500-1796.txt', '20170201014500-342.txt', '20170619153000-151.txt', '20161019054500-265.txt', '20170724164500-264.txt', '20170720171500-811.txt', '20151011121500-510.txt', '20170308213000-281.txt', '20170505211500-1002.txt', '20170201173000-1922.txt', '20170624134500-1547.txt', '20180422164500-255.txt', '20170408053000-1355.txt', '20180417061500-1506.txt', '20170622193000-1329.txt', '20161017031500-1446.txt', '20180302204500-1892.txt', '20151105163000-1374.txt'. '20160314083000-1134.txt'. '20180305053000-789.txt'. '20170501224500-33.txt'. '20180329053000-806.txt'. '20160927013000-83.txt'. '20170612234500-885.txt'. '20
```

In [20]:
```
def find_child_labor_references_in_folder(folder_path, keywords, max_files=10000, encoding='utf-8'):
    child_labor_folder = []

    # Iterate over files in the folder
    file_count = 0
    for filename in os.listdir(folder_path):
        if file_count >= max_files:
            break

        if filename.endswith('.txt'):
            file_path = os.path.join(folder_path, filename)
            with open(file_path, 'r', encoding=encoding, errors='ignore') as file:
                text = file.read().lower()  # Convert text to lowercase for case-insensitive matching

            # Check for keywords in the text
            for keyword in keywords:
                if keyword in text:
                    child_labor_folder.append(file_path)
                    break  # Once any keyword is found, no need to check further for this file

            file_count += 1

    return child_labor_folder

# Example usage
child_labor_folder = find_child_labor_references_in_folder(folder_path, keywords)

if child_labor_folder:
    print("Folder containing files mentioning child labor keywords:", child_labor_folder)
else:
    print("No files mentioning child labor keywords found in the folder.")
```

```
Folder containing files mentioning child labor keywords: ['/Users/subaru/Library/CloudStorage/OneDrive-UniversityofLeeds/Courses/02_Sem2/5403_Creative Coding/Week4/Alex/dataset/20151201211500-2747.txt', '/Users/subaru/Library/CloudStorage/OneDrive-UniversityofLeeds/Courses/02_Sem2/5403_Creative Coding/Week4/Alex/dataset/20170615204500-612.txt', '/Users/subaru/Library/CloudStorage/OneDrive-UniversityofLeeds/Courses/02_Sem2/5403_Creative Coding/Week4/Alex/dataset/20160609040000-247.txt', '/Users/subaru/Library/CloudStorage/OneDrive-UniversityofLeeds/Courses/02_Sem2/5403_Creative Coding/Week4/Alex/dataset/20160525113000-2449.txt', '/Users/subaru/Library/CloudStorage/OneDrive-UniversityofLeeds/Courses/02_Sem2/5403_Creative Coding/Week4/Alex/dataset/20170209214500-1865.txt', '/Users/subaru/Library/CloudStorage/OneDrive-UniversityofLeeds/Courses/02_Sem2/5403_Creative Coding/Week4/Alex/dataset/20170705150000-2474.txt', '/Users/subaru/Library/CloudStorage/OneDrive-UniversityofLeeds/Courses/02_Sem2/5403_Creative Coding/Week4/Alex/dataset/20170526151500-1060.txt', '/Users/subaru/Library/CloudStorage/OneDrive-UniversityofLeeds/Courses/02_Sem2/5403_Creative Coding/Week4/Alex/dataset/20161214084500-1377.txt', '/Users/subaru/Library/CloudStorage/OneDrive-UniversityofLeeds/Courses/02_Sem2/5403_Creative Coding/Week4/Alex/dataset/20180316160000-2105.txt', '/Users/subaru/Library/CloudStorage/OneDrive-UniversityofLeeds/Courses/02_Sem2/5403_Creative Coding/Week4/Alex/dataset/20161203184500-1900.txt', '/Users/subaru/Library/CloudStorage/OneDrive-UniversityofLeeds/Courses/02_Sem2/5403_Creative Coding/Week4/Alex/dataset/20170602001500-2047.txt', '/Users/subaru/Library/CloudStorage/OneDrive-UniversityofLeeds/Courses/02_Sem2/5403_Creative Coding/Week4/Alex/dataset/20170508183000-1830.txt', '/Users/subaru/Library/CloudStorage/OneDrive-UniversityofLeeds/Courses/02_Sem2/5403_Creative Coding/Week4/Alex/dataset/20170516140000-290.txt', '/Users/subaru/Library/CloudStorage/OneDrive-UniversityofLeeds/Courses/02_Sem2/5403_Creative Coding/Week4/Alex/dataset/20180424230000-2526.txt', '/Users/subaru/Library/CloudStorage/OneDrive-UniversityofLeeds/Courses/02_Sem2/5403_Creative Coding/Week4/Alex/dataset/20170424144500-985.txt', '/Users/subaru/Library/CloudStorage/OneDrive-UniversityofLeeds/Courses/02_Sem2/5403_Creative Coding/Week4/Alex/dataset/20170201031500-1681.txt', '/Users/subaru/Library/CloudStorage/OneDrive-UniversityofLeeds/Courses/02_Sem2/5403_Creative Coding/Week4/Alex/dataset/20170601070000-276.txt', '/Users/subaru/Library/CloudStorage/OneDrive-UniversityofLeeds/Courses/02_Sem2/5403_Creative Coding/Week4/Alex/dataset/20161214073000-1302.txt', '/Users/subaru/Library/CloudStorage/OneDrive-UniversityofLeeds/Courses/02_Sem2/5403_Creative Coding/Week4/Alex/dataset/20151221181500-2645.txt', '/Users/subaru/Library/CloudStorage/OneDrive-UniversityofLeeds/Courses/02_Sem2/5403_Creative Coding/Week4/Alex/dataset/20160407164500-1353.txt', '/Users/subaru/Library/CloudStorage/OneDrive-UniversityofLeeds/Courses/02_Sem2/5403_Creative Coding/Week4/Alex/dataset/20170624193000-1048.txt', '/Users/subaru/Library/CloudStorage/OneDrive-UniversityofLeeds/Courses/02_Sem2/5403_Creative Coding/Week4/Alex/dataset/20160114194500-563.txt', '/Users/subaru/Library/CloudStorage/OneDrive-UniversityofLeeds/Courses/02_Sem2/5403_Creative Coding/Week4/Alex/dataset/20180423123000-659.txt', '/Users/subaru/Library/CloudStorage/OneDrive-UniversityofLeeds/Courses/02_Sem2/5403_Creative Coding/Week4/Alex/dataset/20160702130000-1
```

In [21]:
```
#Open the child labour-related text files and show the contents
def process_files(file_paths, encoding='utf-8'):
    for file_path in file_paths:
        # Do something with each file
        print("Processing file:", file_path)
        with open(file_path, 'r', encoding=encoding, errors='ignore') as file:
            text = file.read()
            # Perform further processing or analysis on the file content
            # For example, you can print the content of the file
            print(text)
            print("-" * 50)  # Print a separator between files

# Call process_files with child_labor_folder list
process_files(child_labor_folder, encoding='utf-8')

```

```
Processing file: /Users/subaru/Library/CloudStorage/OneDrive-UniversityofLeeds/Courses/02_Sem2/5403_Creative Coding/Week4/Alex/dataset/20151201211500-2747.txt

woodrow wilson wasn't morally or ethically pure. no president is.

bangor, me

33
clouds

                close

login
logout
subscribe

search for:

                search

--------------------------------------------------
```

In [22]:
```
import pandas as pd
def read_text_files(file_paths):
    data = []
    for file_path in file_paths:
        with open(file_path, 'r', encoding='utf-8') as file:
            content = file.read()
            data.append(content)
    return data

# Assign a dataframe and accommodate contents.
def text_files_to_dataframe(file_paths):
    # Read text files
    texts = read_text_files(file_paths)

    # Create DataFrame
    df = pd.DataFrame({'Text': texts})
    # Optionally, add a column for file names
    df['File_Name'] = [os.path.basename(file_path) for file_path in file_paths]
    return df


# Create DataFrame
df = text_files_to_dataframe(child_labor_folder)

# Display DataFrame
print(df)

```

```
                                                  Text  \
0    \nwoodrow wilson wasn't morally or ethically p...
1    hp is turning trash into printer cartridgeseng...
2    \npakistan's censor has a thing or two to say ...
3    president obama talks about human rights, bett...
4    what we uncovered about ivanka trump's fashion...
..                                                 ...
924  migrant construction workers' children in thai...
925  top aid groups call on obama to help refugees×...
926  \nis your work killing you? \n\nskip to the co...
927  \nyou can't have financial inclusion without d...
928  \nhershey to invest $0.5bn by 2030 for sustain...

             File_Name
0    20151201211500-2747.txt
1     20170615204500-612.txt
2     20160609040000-247.txt
3    20160525113000-2449.txt
4    20170209214500-1865.txt
..                       ...
924   20180329113000-320.txt
925  20160920184500-1590.txt
926  20180424141500-2753.txt
927  20160715114500-1237.txt
928   20180405150000-996.txt

[929 rows x 2 columns]
```

In [244]:
```
## Search for specific articles
test_keywords = 'United States'
test = df[df['Text'].str.contains(test_keywords, case=False)]
test
```

Out[244]:

	Text	File_Name
0	\nwoodrow wilson wasn't morally or ethically p...	20151201211500-2747.txt
3	president obama talks about human rights, bett...	20160525113000-2449.txt
5	\nthai pm halts migrant labor law after protes...	20170705150000-2474.txt
7	\neducation attainment: muslims, hindus lag fa...	20161214084500-1377.txt
8	\na colorful bus brings books and joy to afgha...	20180316160000-2105.txt
...	...	...
916	top 8 countries with most horrific human right...	20180423150000-1619.txt
919	\nhuman rights archives - u.s. embassy in tuni...	20170612120000-2284.txt
922	betsy devos: religion and profit in the war on...	20161213081500-1022.txt
923	\nstudents pulled between work and school in p...	20161130143000-972.txt
926	\nis your work killing you? \n\nskip to the co...	20180424141500-2753.txt

301 rows × 2 columns

In [254]:
```
from IPython.core.interactiveshell import InteractiveShell
InteractiveShell.ast_node_interactivity = "all"
```

In [265]:
```
df[df.index==65]
```

Out[265]:

	Text	File_Name
65	\na loss of nerve | commonweal magazine\n\nmai...	20160117094500-947.txt

In [31]:
```
df.iloc[10,0]
```

Out[31]:
'guatemala rescues 22 children from forced labor: police | reuters\n\nskip to main contentexclusive news, data and analytics for financial market professionalslearn more aboutrefinitivreuters homeworldbusinessmarketssustainabilitylegalbreakingviewstechnologyinvestigationsmoremy viewregisterworldguatemala rescues 22 children from forced labor: policeby anastasia moloneyjune 1, 20174:59 pm utcupdated agobogota (thomson reuters foundation) - guatemalan police have rescued 22 children - some working 16-hour days making corn tortillas - as part of a crackdown on widespread child labor in the central american nation, according to public prosecutors.police arrested six people on charges of child labor on wednesday, following 37 raids on shops and food stalls in the capital guatemala city.the children rescued were aged 14 and 17, and most came from impoverished indigenous communities in rural areas.children were forced to work seven days a week in unhealthy conditions, and with little pay and food, prosecutors said.the police raids aimed to target "those whose take advantage of a child\'s vulnerability to get cheap labor," the public prosecutor\'s office said in a statement on wednesday.poverty drives child labor in guatemala.about half of guatemala\'s 15 million people live in poverty and the country has one of the world\'s highest rates of chronic malnutrition, mostly affecting the children of indigenous communities.to put food on the table, some parents encourage their children to quit school and work in the capital, leaving them open to sexual exploitation and forced labor, experts say.nearly 415,000 children, some as young as seven, work in guatemala, according to the u.s. government.most child labor exists in agriculture, along with domestic work, forced begging and garbage scavenging.it is common to see children working day and night as shoe shiners and street vendors on the capital\'s streets.worldwide there are 168 million children in child labor, according to the international labour organization.the united nations has a global goal to end all child labor by 2025 and to eradicate forced labor and slavery by 2030.reporting by anastasia moloney @anastasiabogota, editing by lyndsay griffiths. please credit the thomson reuters foundation, the charitable arm of thomson reuters, that covers humanitarian news, women\'s rights, trafficking, property rights, climate change and resilience. visit news.trust.orgour standards: the thomson reuters trust principles. acquire licensing rights, opens new tabread nextunited statescategoryinfluential koch group endorses haley\'s 2024 republican presidential bid6:53 pm utcarticle with videoeuropecategoryukraine says wife of spymaster budanov was poisoned7:03 pm utc · updated agoworldcategoryiran finalises deal to buy russian fighter jets - tasnim9:04 am utcarticle with galleryeuropecategoryeu court says public employees may be barred from wearing head scarf1:35 pm utcmore from reutersworldweek after shock wilders win, dutch government talks set to begineuropecategory · november 28, 2023 · 11:53 am utctalks to form a new dutch government were set to start on tuesday, almost a week after the upset election victory of anti-islam populist geert wilders, with a former government minister picked to sound out workable coalitions.article with galleryeuropecategorygreece, britain trade blame over cancelled meeting in parthenon marbles dispute5:49 pm utcarticle with galleryeuropecategorygerman authorities urge people to stay home amid deadly winter weather10:51 am utcunited statescategoryembattled republican george santos faces fresh move to oust him from house7:15 pm utc · updated agoarticle with videoworldcategoryisrael-hamas war: answering readers' questions4:59 pm utcsite indexbrowseworldbusinessmarketssustainabilitylegalbreakingviewstechnologyinvestigations, opens new tabsportsciencelifestyleabout reutersabout reuters, opens new tabcareers, opens new tabreuters news agency, opens new tabbrand attribution guidelines, opens new tabreuters leadership, opens new tabreuters fact check, opens new tabreuters diversity report, opens new tabstay informeddownload the app (ios), opens new tabdownload the app (android), opens new tabnewsletters, opens new tabinformation you can trustreuters, the news and media division of thomson reuters, is the world's largest multimedia news provider, reaching billions of people worldwide every day. reuters provides business, financial, national and international news to professionals via desktop terminals, the world\'s media organizations, industry events and directly to consumers.follow usthomson reuters productswestlaw, opens new tabbuild the strongest argument relying on authoritative content, attorney-editor expertise, and industry defining technology.onesource, opens new tabthe most comprehensive solution to manage all your complex and ever-expanding tax and compliance needs.checkpoint, opens new tabthe industry leader for online information for tax, accounting and finance professionals.lseg productsworkspace, opens new tab access unmatched financial data, news and content in a highly-customised workflow experience on desktop, web and mobile.data catalogue, opens new tab browse an unrivalled portfolio of real-time and historical market data and insights from worldwide sources and experts.world-check, opens new tabscreen for heightened risk individual and entities globally to help uncover hidden risks in business relationships and human networks.advertise with us, opens new tabadvertising guidelines, opens new tabcoupons, opens new tabacquire licensing rights, opens new tabcookies, opens new tabterms of use, opens new tabprivacy, opens new tabdigital accessibility, opens new tabcorrections, opens new tabsite feedback, opens new taball quotes delayed a minimum of 15 minutes. see here for a complete list of exchanges and delays.© 2023 reuters. all rights reserved'

In [30]:
```
df_text_file = df['File_Name']
df_text_file.to_csv('df_text_file.csv', index=False, encoding='utf-8')
```

Step3: Analysis with ChatGPT

In [16]:
```
import os
import openai
from openai import OpenAI
from openai.types import Completion, CompletionChoice, CompletionUsage

API_KEY= "XXXXX"
```

3-1: Create function to identify Companies

In [72]:
```
def chat_gpt_companies(text, API_KEY):
    client = OpenAI(api_key = API_KEY)
    response = client.chat.completions.create(
        messages=[

            {
                "role": "system",
                "content": f"""Can you find the compaines that are mentioned in relation to child labour exploitation?
                Here is the text:'{text}'.
                Please provide a list of company names, without any additional text or sentences.
                If there are no relevant company names found, please return 'None'.
                Here is an example of your message: 'Microsoft, Alphabet'.
                """
            },
            {
                "role": "user",
                "content": text
            }
            ],
            model = "gpt-3.5-turbo"
    )

    return (response.choices[0].message.content)
```

3-2: Create function to identify Commodities

In [73]:
```
## function to extract commodity info using chat gpt api
def chat_gpt_commodities(text, API_KEY):
    client = OpenAI(api_key = API_KEY)
    response = client.chat.completions.create(
        messages=[

            {
                "role": "system",
                "content": f"""Can you find the commodities that are mentioned in relation to child labour exploitation?
                Here is the text: '{text}'.
                Please provide a list of commodity names, without any additional text or sentences, using only the names in the commodity list: '{commodity_labour}'.
                If there are no relevant commodity names found, please return 'None'.
                Here is an example of your message: 'Jewelry, Apparel'.
                """
            },
            {
                "role": "user",
                "content": text
            }
            ],
            model = "gpt-3.5-turbo"
    )
    return (response.choices[0].message.content)


```

3-3: Create function to identify Countries

In [74]:
```
## function to extract country info using chat gpt api
def chat_gpt_countries(text, API_KEY):
    client = OpenAI(api_key = API_KEY)
    response = client.chat.completions.create(
        messages=[

            {
                "role": "system",
                "content": f"""Can you find the countries that are mentioned in relation to child labour exploitation?
                Here is the text: '{text}'.
                Please provide a list of country names, without any additional text or sentences.
                If there are no relevant country names found, please return 'None'.
                Here is an example of your message: 'United States, Indonesia'.
                """
            },
            {
                "role": "user",
                "content": text
            }
            ],
            model = "gpt-3.5-turbo"
    )

    return (response.choices[0].message.content)
```

3-4: Run all the ChatGPT functions and Store returned result into DataFrame

CAUTION!!!: Please DO NOT RUN the following chunk as it consumes a lot of tokens.

In [83]:
```
# !CAUTION! Please DO NOT RUN this step as it consumes a lot of tokens.######
##################################

import pandas as pd

# Assuming you have created an empty DataFrame named output_df
#output_df = pd.DataFrame(columns=['Companies', 'Commodities', 'Countries', 'Additional_Info'])
output_df = pd.DataFrame(columns=['Companies', 'Commodities', 'Countries'])

# Assuming you have a DataFrame named input_df with a column named 'text' containing the text data
start_index = 0
for index, row in df.iloc[start_index:].iterrows():
    text = row["Text"]
    try:
        output1 = chat_gpt_companies(text, API_KEY)
        output2 = chat_gpt_commodities(text, API_KEY)
        output3 = chat_gpt_countries(text, API_KEY)
        print("Index: {}, Companies: {} Commoditites: {} Countries: {}".format(index,output1,output2,output3))
        output_df.loc[index] = [output1,output2,output3]
    except Exception as e:
        print(f"Error processing text index: {index}.")
        continue


# Now output_df contains the combined outputs
print(output_df)

```

```
Index: 0, Companies: None Commoditites: None Countries: None
Index: 1, Companies: None Commoditites: Recycled materials Countries: Haiti, Honduras
Index: 2, Companies: None Commoditites: Children, Child Labor, Child Abuse, Contraceptives. Countries: Pakistan
Index: 3, Companies: None Commoditites: None Countries: United States, Vietnam, China
Index: 4, Companies: G-III Apparel Group Commoditites: Apparel, Textiles, Garments, Fashion Countries: Indonesia, China, Vietnam
Index: 5, Companies: None Commoditites: Fisheries, Construction, Agriculture Countries: Myanmar, Laos, Cambodia
Index: 6, Companies: None Commoditites: Textiles, Apparel Countries: Kyrgyzstan
Index: 7, Companies: None Commoditites: None Countries: United States, Israel, India, Nepal, Bangladesh, Sub-Saharan Africa, Europe, Middle East
Index: 8, Companies: None Commoditites: Books Countries: Afghanistan
Error processing text index: 9.
Index: 10, Companies: None Commoditites: Corn, Food Processing Products, Agricultural Produce Countries: Guatemala, United States
Index: 11, Companies: None Commoditites: 'Child labor' Countries: Mexico, United States
Index: 12, Companies: None Commoditites: None Countries: Pakistan
Index: 13, Companies: None Commoditites: None Countries: United States, Central America
Index: 14, Companies: Nike, Gap. Commoditites: Cotton, Textiles, Apparel, Garments Countries: Bangladesh, Uzbekistan, India
Index: 15, Companies: None Commoditites: Ores, Gold, Textiles, Apparel, Electronics Countries: Brazil, Canada
Index: 16, Companies: Disney Commoditites: Newspaper Countries: United States, New York
Index: 17, Companies: None Commoditites: None Countries: Afghanistan, Pakistan
Index: 18, Companies: Thai Union, Thai Frozen Foods Association Commoditites: Shrimp, Seafood, Migrant laborers Countries: Thailand, Myanmar, United States, Europe, Australia, Indonesia
```

3-5: Add summary column to DataFrame that contains involved companies, commodities and countries in each case

In [148]:
```
output_df['comb'] = '"Case'+ output_df.index.astype(str) + '":<<' + 'Companies: ' + output_df['Companies'].astype(str) + '. Commodities: ' +  output_df['Commodities'].astype(str) + '. Countries: ' +  output_
output_df
```

Out[148]:

	Companies	Commodities	Countries	comb
0	None	None	None	"Case0":<<Companies: None. Commodities: None. ...
1	None	Recycled materials	Haiti, Honduras	"Case1":<<Companies: None. Commodities: Recycl...
2	None	Children, Child Labor, Child Abuse, Contracept...	Pakistan	"Case2":<<Companies: None. Commodities: Childr...
3	None	None	United States, Vietnam, China	"Case3":<<Companies: None. Commodities: None. ...
4	G-III Apparel Group	Apparel, Textiles, Garments, Fashion	Indonesia, China, Vietnam	"Case4":<<Companies: G-III Apparel Group. Comm...
...	...	...	...	...
924	None	None	Cambodia, Myanmar, Laos	"Case924":<<Companies: None. Commodities: None...
925	None.	None.	Syria, South Sudan, Yemen, Burundi, Central Af...	"Case925":<<Companies: None.. Commodities: Non...
926	None	None	United States, United Kingdom	"Case926":<<Companies: None. Commodities: None...
927	None	None	None	"Case927":<<Companies: None. Commodities: None...
928	hershey	Cocoa, Food processing products, Agricultural ...	Côte d'Ivoire, Ghana	"Case928":<<Companies: hershey. Commodities: C...

908 rows × 4 columns

3-6: Export the result as CSV file.

In [149]:
```
output_df.to_csv('output_data2.csv', index=False)
```

In [150]:
```
output_df
```

Out[150]:

	Companies	Commodities	Countries	comb
0	None	None	None	"Case0":<<Companies: None. Commodities: None. ...
1	None	Recycled materials	Haiti, Honduras	"Case1":<<Companies: None. Commodities: Recycl...
2	None	Children, Child Labor, Child Abuse, Contracept...	Pakistan	"Case2":<<Companies: None. Commodities: Childr...
3	None	None	United States, Vietnam, China	"Case3":<<Companies: None. Commodities: None. ...
4	G-III Apparel Group	Apparel, Textiles, Garments, Fashion	Indonesia, China, Vietnam	"Case4":<<Companies: G-III Apparel Group. Comm...
...	...	...	...	...
924	None	None	Cambodia, Myanmar, Laos	"Case924":<<Companies: None. Commodities: None...
925	None.	None.	Syria, South Sudan, Yemen, Burundi, Central Af...	"Case925":<<Companies: None.. Commodities: Non...
926	None	None	United States, United Kingdom	"Case926":<<Companies: None. Commodities: None...
927	None	None	None	"Case927":<<Companies: None. Commodities: None...
928	hershey	Cocoa, Food processing products, Agricultural ...	Côte d'Ivoire, Ghana	"Case928":<<Companies: hershey. Commodities: C...

908 rows × 4 columns

In [151]:
```
output_df['comb'][30]
```

Out[151]: '"Case30":<<Companies: None. Commodities: cocoa. Countries: Ivory Coast>>
'

Step4: Summary Analysis with ChatGPT

4-1: Data preparation for feeding data into ChatGPT

In [19]:
```
# Extracting column values into a list
list_companies = output_df['Companies'].tolist()
string_companies = ', '.join(list_companies)

# Extracting column values into a list
list_commodities = output_df['Commodities'].tolist()
string_commodities = ', '.join(list_commodities)

# Extracting column values into a list
list_countries = output_df['Countries'].tolist()
string_countries  = ', '.join(list_countries)

# # Extracting column values into a list
list_comb = output_df['comb'].tolist()
string_comb = ', '.join(list_comb)
print(string_comb)

```

```
---------------------------------------------------------------------------
NameError                 Traceback (most recent call last)
Cell In[19], line 4
      1 # !CAUTION! Please DO NOT RUN this step as it consumes a lot of tokens.######
      2
      3 # Extracting column values into a list
----> 4 list_companies = output_df['Companies'].tolist()
      5 string_companies = ', '.join(list_companies)
      7 # Extracting column values into a list

NameError: name 'output_df' is not defined
```

In [154]:
```
string_companies
```

Out[154]:
"None, None, None, None, G-III Apparel Group, None, None, None, None, None, None, None, None, Nike, Gap., None, Disney, None, Thai Union, Thai Frozen Foods Association, None, Nike, Puma, Belle International, Adidas, None, None., None, Driscoll's, H&M, Gap, Inditex, Walmart, C&A., None, Reynolds American, None, None, ExxonMobil, Arch Coal, BP, Coca-Cola., None, None, None, None, Wells Fargo, Tesla, Agriprocessors, Wendy's, Mcdonald's, Walmart, Taco Bell, Pizza Hut, KFC, Yum! Brands., None, None, Apple Inc, Sony Corp, Volvo, None, None, None, Hershey, Mars, Nestle, ADM Cocoa, Guittard Chocolate Company, Godiva, Chocolates by Bernard Callebaut, Fowler's Chocolate, Kraft, See's Candies., ExxonMobil, Scott Pruitt, None, None, None, None, Maruha Nichiro, Nippon Suisan Kaisha, Thai Union, Dongwon Industries, Marine Harvest ASA, Cermaq, Nutreco Unit Skretting, Cargill Aqua Nutrition, Austevoll, Charoen Pokphand Foods, Kyokuyo, Pacific Andes, Pescanova, Trident, None, Accenture, Child in Need Institute (CINI), None, Save the Children, UNICEF, None, None, None, None, None, None, None, None., Philip Morris International Inc, Djarum Group, PT Gudang Garam Tbk., None, None, None, None, None, None, None, None, None, None, h&m, Marks & Spencer, Tesco, Primark, Gap, adidas, c&a, carrefour, disney, gap, h&m, inditex, kering, li & fung, nike, tesco, Mars, Hershey, Barry Callebaut, Ferrero, Nestle, Lindt, Mondelez, Ritter Sport, Olam, Cargill., Driscoll's, None, None., None, Apple, Huayou, Congo Dongfang Mining, None, Starbucks, Publix, Morgan & Morgan, Shavitz Law Group, None, None, None, None, None, None, None, None, None, None, None, Tazreen Fashions, None, None, Walt Disney, DLA Piper, None, Human Rights Watch, Greenpeace India, Ford Foundation, Eder Flag Manufacturing Co., Trump, Donald J. Trump Collection, Walmart, None, None, None, None, None, None, deepgreen, Glencore, Maersk, Coca Cola., Driscoll's, None, Save the Children, None, None, None, None, None, None, Hershey's, Nestle, Mondelez, Nestle, Balsu, Olam-Progida, None, None, None, None, None, National Rifle Association, None, None, Doosan, Winrock International, None, None, None, None, None, None, None, Nestle, Archer-Daniels-Midland, Cargill, None, None, None, None, None, Natural Fruit, Finnwatch, None, None, None, None, None, None., None, None, None, Japan Tobacco International, Nonprofit group, None, None, None, None, None, None, None, None, None, None, Accenture, None., None, Cargill, Nestle, None, None, None, None, None, None, None, None, Walmart, Nestle, Cargill, None, None, None, Seaside Company, Santa Cruz Beach Boardwalk., Mars, Hershey, Barry Callebaut, Ferrero, Nestle, Lindt, Mondelez, Ritter Sport, Olam, Cargill, None, None, None, None, None, British American Tobacco, None, None, None, alta garcia, sodexo, None, Carrier, Rexnord, None, None, None, None, qsol, None, None, None, None, None, None, None, None, SawitWatch, None, None, Apple Inc, Sony Corp, Volvo, Zhejiang Huayou Cobalt Co Ltd, BMW, None, Cargill, Nestle, ogletree, deakins, nash, smoak & stewart, I could not find any companies mentioned in relation to child labour exploitation in the text provided., Volkswagen, BMW, Maruti Suzuki, Hyundai, Honda, Audi, Mercedes Benz, Renault, Mahindra, Tata Motors, Fujian Kuncai Material Technology Co Ltd, Lush, L'oreal., Paragon Contractors, None, None, None, None, Hardee's, Carl's Jr., Pepsico, Indofood, Metalor, Produits Artistiques Métaux Précieux (PAMP), Kaloti Jewellery International, Emirates Gold, Kundan, Rand Refinery, None, None, C&A Foundation, Patagonia, None, None, None, None, Samsung, Apple., None, None, None, None., None, H&M, Zara (Inditex), VFC (Lee and Wrangler), Mango, None, None, None, Freedom Fund, Tiffany and Company, Bulgari, Cartier, Pandora, Signet, None, United Farm Workers, Bacardi, None, None, None, None, None, None, None, None., None, Walmart, Sam's Club, None., None, None, None, Apple, Samsung, Sony, None, Kellogg's, Wilmar, Siemens, Puma Energy, LS Cable & System, Uber, None, None, None, None, None., IBM, None, None, coca-cola, bitfury, emercoin, None, None, None, None, Wendy's, None, None, None, None, Unicef, Millicom, H&M, Signet Jewelers, Zales, Kay Jewelers, Jared, Shane Company, Helzberg Diamonds, Luxottica, None, None, None, None, United Technologies, Carrier, Cardone, Dematic, Hershey., None, None, de beers, PricewaterhouseCoopers, None, None, Catholic Relief Services, None, None, None, None, ferrero, None, None, None, Nike, Apple, Apple, Nike, None, None, Martin Luther King, None, None, None, None, None, Apple, Microsoft, Tesla, Samsung, Next, New Look, Asos, C&A, H&M, Kik, LC Waikiki, Primark, Otto Group, Esprit, Gap, Mothercare, Tesco, Tchibo, White Stuff, Adidas, Debenhams, Puma, Arcadia, Burberry, S.Oliver, Supergroup, VF Corp, Walmart, None, None, None, None, None, British American Tobacco, Alliance One International, Universal Leaf Tobacco, China Labor Watch, Walt Disney Co., Samsung, Apple Inc., None, None, None, Paragon Contractors, None, None, None, None, Hobby Lobby, Apple, None, None, None, None, None, Marks & Spencer, ASOS, Mango, Zara, Inditex, Human Rights Watch, Walmart, Costco, None, None, None, H&M, Inditex, C&A, PVH, None, None, None, None, None., Apple, Tesla, Apple Inc, Sony Corp, Volvo, None, None, None, None, None, None, None, None, None, cez, korea electric power (kepco), norges bank investment management (nbim), None, None, None, None, Barry Callebaut., Hershey's, Apple, Samsung, Sony, Volkswagen, None, Telenor, Tele2, Telia Company, Clas Ohlson, H&M, Trelleborg., None, None, Thomson Reuters Foundation, Pacific Links Foundation, Liberty Asia, Fair Trade., None, Cargill, Monsanto, Save the Children, None, None, None, None, None, None, None, Mondelez, Kellogg, Nestlé, Unilever, Danone, None., Driscoll's, Amalgamated Plantations Private Limited (APPL), None, Foxconn, ison, None, De Beers, None, Foxconn., None, None, ferrero, dunkin' donuts, None, None, None., None, None, None, None, Apple, Samsung, Congo Dongfang International Mining, Zhejiang Huayou Cobalt, None, Fujian Kuncai Material Technology Co Ltd, Merck KGaA, L'Oreal., Nestlé, Cargill, Archer Daniels Midland, Indorama Corporation, None, None, None, None, None, None, None, Mars, Hershey, Ferrero, None, dorae inc, Aethel Partners, None, Kellogg's, Lockheed Martin, None, Trillium Farms, None, None, Thai Union, Walmart, None, None, None, Nestle, Archer Daniels Midland Co, Cargill Inc, None., None, None, Zhejiang Huayou Cobalt Company, None, None, None, None, None, Department of State, Wal-Mart, Whole Foods, Red Lobster, Thai Union, Volcom, Apple, None, None, None, None, Target Corp., Nike Inc., None, None, None, None, None, Meta, Facebook, Instagram, NTR Metals, None, None, Ben & Jerry's, Rishi Tea, H&M, Victoria's Secret, Hershey's, Forever 21, Walmart, Gap, Apple., None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, Boston Consulting Group, Accenture, KPMG, BMW Foundation, HP, Google, Microsoft, Ericsson, Western Union, Mastercard, Facebook, Uber, None, The Acton Institute., HP Inc., None, None, None, None, None, None, None., None, ecobalt solutions, Jervois Global Limited, LGCLF, PCFFF, SAMSUNG, LG CHEMICAL, PANASONIC, FORTUNE MINERALS, TIGER RESOURCES, LUNDIN MINING, CRUZ COBALT CORP, FREEPORT-MCMORAN, KATANGA MINING, GLENCORE, BARRA RESOURCES, Cargill, Nestle, None, None, None, None, None, None, None, None, None, None, None, None, European Adoption Consultants, None, None, Thai Union, SAP Ariba, No nasties, Do U Speak Green, None, None, Foxconn, Formosa, Katanga Mining Limited, Fortune Minerals Limited, EcoCobalt Solutions, Bankers Cobalt Corp., None, None, None, None., None, None, None, None, None, Accenture, Child in Need Institute (CINI), None, None., Mars, Hershey, Barry Callebaut, Ferrero, Nestle, Lindt, Mondelez, Ritter Sport, Olam, Cargill., None, None, Timberland, HP, Gap, C&A, H&M., None, None, Mars, Hershey, Ferrero, Nestle, None, None, Nestle, Ferrero, Mars, None., None, None, None, None, None, None, None, save the children, None, coca-cola, bitfury group, emercoin, blockchain trust accelerator, None, None, ioi, None, Paragon Contractors Corp., None, None, None, None, None, None, None, None, Apple, Microsoft, Tesla, Samsung., None, None, None, None, None, None, None, None, None., None, Amnesty International, Unilever, Procter & Gamble, Kellogg's, Nestle, None, None, Samsung, Huawei, Microsoft., None, None, None, None, None, Paragon Contractors., None., PepsiCo, Indofood, Hobby Lobby., None, None, Timberland, CAEI Group, Driscoll's, None, None, None, c&a, hennes & mauritz (h&m), ikea, samsung, None, None, Apple, Zhejiang Huayou Cobalt Company, None, None, None, Monsanto, Bayer, Reuters, None, Costco, Nestle, Oxfam, None, Save the Children, None., None, None, None, Rexnord, Carrier, United Technologies, None, Sainsbury, Black Dinah Chocolatiers, Madecasse, None, Nestlé, Save the Children, None, American Express, Cisco Systems, Walmart, Boeing, IBM, Microsoft, Coca-Cola, Merck, McDonald's, Apple., None, None, None, None, None, Human Rights Watch, None, Next, New Look, ASOS, C&A, H&M, Kik, LC Waikiki, Primark, Otto Group, Esprit, Gap, Mothercare, Tesco, Tchibo, White Stuff, Adidas, Debenhams, Puma, Inditex, Arcadia, Burberry, S.Oliver, SuperGroup, VF Corp, Walmart, Gerry Weber, Lidl, Mexx, New Yorker, River Island, Sainsbury's., Dassault Aviation, Safran, Thales, None, None, Smart Guy, Short-Lived Smart Guy, None, Compassion International Ghana, None, Costco, Whole Foods, Driscoll, None, None, None, None, None, None, None, None, None, Girls not Brides, Fashion Revolution, Labor Behind the Label, None, None, Verizon, Communications Workers of America (CWA), International Brotherhood of Electrical Workers (Ibew), None, None, None, None, None, None, None, None, None, None, None, Ivanka Trump, Nordstrom, Neiman Marcus, Burlington, Belk, Kmart, Sears, Wegmans, Nike, Apple, Cracker Barrel, Domino's, Chik-fil-a, Hobby Lobby, Starbucks, DeBeers, Walmart., None, None, None, None, None, None, Altria Group, Alliance One International (AOI), British American Tobacco (BAT), China National Tobacco Corporation, Intercontinental Leaf Tobacco (ILT), Contraf Nicotex Tobacco GmbH (CNT), Imperial Tobacco (Imperial), Japan Tobacco International (JTI), Rift Valley Corporation/Northern Tobacco (NT), Premium Tobacco Group (Premium)/Premium Tobacco Zimbabwe (PTZ), Philip Morris International (PMI), Universal Corporation, Nestle, Unilever, Procter & Gamble, Colgate-Palmolive., None., H&M, National Public Radio, None., First Cobalt Corp, US Cobalt Inc, Agnico Eagle Mines Ltd, Sherritt International Corp, Vale SA, Cobalt 27 Capital Corp., None, None, Hershey, Vauxhall, Volkswagen, BMW, L'Oréal, Estée Lauder., None, None, Foxconn., None, None, None, None, None, None, None, None, coca-cola, None, None, None, None, None, Barry Callebaut, Blommer Chocolate Company, Cargill, Cemoi, Ecom, Ferrero, The Hershey Company, Mars Incorporated, Mondel?z International, Nestlé, Olam, Touton, None, None, None, None, Amal Clooney, Kailash Satyarthi, None, None, None, Apple, Target, Nike, None, None, None, None, None, None, None, None, None, None, None, Cacao Barry, Adidas, Microsoft, None, None, None, None, Nestlé, Archer-Daniels-Midland, Cargill, None, Procter & Gamble, None, None, None, None, None, None, None, H&M, Primark, Tesco, None, None, None, None, None, None, None, grupo palmas, alicorp, Mattel, Amazon, None, None, None, None, None, None, Freeport-McMoRan, Glencore, China Molybdenum, Samsung, None, None, None, None, None, Marc Fisher, Nine West, Karl Lagerfeld, Kendall+Kylie, Easy Spirit, World Food Programme, European Union, None, Applied DNA Sciences, None, None, None, None, None., None, None, None, None, None, None, None, None, None, None., None, None, None, None, None, None, None, None, None, None, None, Huajian Group, Ivanka Trump, Marc Fisher, Nine West, Easy Spirit, None, None, None, None, None, GM, Renault-Nissan, Fiat Chrysler, Volkswagen, Daimler, Tesla, None, None, None, Apple, None, Warren Jeffs, None, Snow, Christensen and Martineau, None, None, None, Gap, H&M, Forever 21, Wendy's, None, None, None, None, H&M, Inditex, Louisiana Association of Business and Industry (LABI), None, Phillip Morris, Altria, British American Tobacco, None, None., None, None, None, None, coca-cola, humanity united, None, None, None, None, Driscoll's berries, None, None, None, None, Walmart, Dick's Sporting Goods, Kroger, BlackRock, REI, Timberland, None, None, None, None, None, None, None, Apple, None, de beers, None, None, None, None, None, None, None, None, None, Huajian Group, Walt Disney Co., Samsung, Apple Inc., Ivanka Trump, Marc Fisher, Nine West, Easy Spirit, None, None, Macy's, Kohl's, Army and Air Force Exchange Service (AAFES), Apple, HP, Samsung SDI, Sony, Dell, Foxconn, Ford Motors, Tesla, LG Chem, Umicore., None, None, PepsiCo, adidas, Kenneth Cole, None, None, None, None, H&M, Primark, Tesco, Huajian Group, None, Gildan Activewear, Nike, None, None, Abby Lee Dance Company, Dance Moms, None, China Labor Watch, Walt Disney Co., Marc Fisher, Nine West, None, None, None, None, Levi Strauss, Driscoll, Debeers, Anglo American, Diamond Development Initiative (DDI), None, None, None, None, McDonald's, None, None, Cargill, Conagra, h&m, Marks & Spencer, Tesco, Primark, Gap, None, None, Indofood, PepsiCo, None, Pepsico, Nestlé, Unilever, Kellogg's, McDonald's, Mars, Procter & Gamble, None, None, Hershey., None, Center for Biological Diversity, None, None, Wendy's, McDonald's, Walmart, Taco Bell, Pizza Hut, KFC, CIW, None., None, None, None, None, None, None., None, None, None, Amway, The Citizens Foundation, Developments in Literacy, Alif Ailaan, None, None., None, None, hershey"

4-2: Summary analysis for top 20 companies with ChatGPT

In [156]:
```
#Companies
client = OpenAI(api_key = API_KEY)
response = client.chat.completions.create(
    messages=[

        {
          "role": "system",
          "content": """
          Your role is to summarise the list of companies involved in child labour.
          Here is the list: """ + string_companies
        },
        {
          "role": "user",
          "content": """
          Given the list provided, please rank the top 20 companies that are most likely to be found in relation to child exploitation.
          Please ignore 'None' in the list. """
        }
        ],
        model = "gpt-3.5-turbo"
    )

ranked_companies = response.choices[0].message.content
print(ranked_companies)
```

Based on the list provided, the top 20 companies most likely to be found in relation to child exploitation are as follows:

1. Nestle
2. Cargill
3. Walmart
4. Nike
5. Apple
6. Hershey
7. Mars
8. Ferrero
9. Driscoll's
10. H&M
11. Inditex
12. Gap
13. Pepsico
14. Procter & Gamble
15. McDonald's
16. Coca-Cola
17. Samsung
18. Amazon
19. Starbucks
20. Disney

These companies have appeared most frequently in the list in relation to child exploitation.

4-3: Summary analysis for top 20 commodities with ChatGPT

In [157]:
```
#Commodities
client = OpenAI(api_key = API_KEY)
response = client.chat.completions.create(
  messages=[

    {
      "role": "system",
      "content": """
      Your role is to summarise the list of commodities involved in child labour.
      Here is the list: """ + string_commodities
    },
    {
      "role": "user",
      "content": """
      Given the list provided, please rank the top 20 commodities that are most likely to be found in relation to child exploitation.
      Please ignore 'None' in the list. """
    }
    ],
    model = "gpt-3.5-turbo"
  )

ranked_commodities = response.choices[0].message.content
print(ranked_commodities)
```

Based on the list of commodities mentioned in relation to child labor exploitation, the top 20 commodities likely to be found in such situations are:
1. Textiles
2. Apparel
3. Footwear
4. Toys
5. Handicrafts
6. Electronics
7. Garments
8. Furniture
9. Ores
10. Bricks
11. Carpentry
12. Carpets
13. Paper products
14. Agricultural produce
15. Medical supplies
16. Packaging materials
17. Metal products
18. Building materials
19. Leather goods
20. Cosmetics

4-4: Summary analysis for top 20 countries with ChatGPT

In [158]:
```
#Countries
client = OpenAI(api_key = API_KEY)
response = client.chat.completions.create(
  messages=[

    {
      "role": "system",
      "content": """
      Your role is to summarise the list of countries involved in child labour.
      Here is the list: """ + string_countries
    },
    {
      "role": "user",
      "content": """
      Given the list provided, please rank the top 20 countries that are most likely to be found in relation to child exploitation.
      Please ignore 'None' in the list. """
    }
    ],
    model = "gpt-3.5-turbo"
  )

ranked_countries = response.choices[0].message.content
print(ranked_countries)
```

Based on the extensive list provided, the top 20 countries that are most likely to be found in relation to child exploitation are:

1. India
2. United States
3. Pakistan
4. Bangladesh
5. Syria
6. Afghanistan
7. Iraq
8. Nigeria
9. Mexico
10. Indonesia
11. Democratic Republic of Congo
12. Vietnam
13. Yemen
14. Thailand
15. Russia
16. China
17. Lebanon
18. Sudan
19. Mali
20. Ghana

4-5: Export the results

In [159]:
```
# File path
file_path = "ranked_output2.txt"

# Open the file in write mode
with open(file_path, "w") as file:
    # Write the strings to the file
    file.write("Worst Countries are"+ ranked_countries + "\n")
    file.write("Most common companies are" +ranked_companies + "\n")
    file.write("Most common commodities are" +ranked_commodities + "\n")

print("File created successfully.")

```

File created successfully.

Step5: Visualisation

In [53]:
```
output_df = pd.read_csv("output_data2.csv")
output_df
```

Out[53]:

	Companies	Commodities	Countries	comb
0	None	None	None	"Case0":<<Companies: None. Commodities: None. ...
1	None	Recycled materials	Haiti, Honduras	"Case1":<<Companies: None. Commodities: Recycl...
2	None	Children, Child Labor, Child Abuse, Contracept...	Pakistan	"Case2":<<Companies: None. Commodities: Childr...
3	None	None	United States, Vietnam, China	"Case3":<<Companies: None. Commodities: None. ...
4	G-III Apparel Group	Apparel, Textiles, Garments, Fashion	Indonesia, China, Vietnam	"Case4":<<Companies: G-III Apparel Group. Comm...
...	...	...	...	...
903	None	None	Cambodia, Myanmar, Laos	"Case924":<<Companies: None. Commodities: None...
904	None.	None.	Syria, South Sudan, Yemen, Burundi, Central Af...	"Case925":<<Companies: None.. Commodities: Non...
905	None	None	United States, United Kingdom	"Case926":<<Companies: None. Commodities: None...
906	None	None	None	"Case927":<<Companies: None. Commodities: None...
907	hershey	Cocoa, Food processing products, Agricultural ...	Côte d'Ivoire, Ghana	"Case928":<<Companies: hershey. Commodities: C...

908 rows × 4 columns

5-1: Countries

In [126]:
```
output_df['Countries'] = output_df['Countries'].str.replace('.', '')
output_df
```

/var/folders/ct/fb9zr96x3g1dxcqhdytdzv5h0000gn/T/ipykernel_3558/3623789606.py:1: FutureWarning:

The default value of regex will change from True to False in a future version. In addition, single character regular expressions will *not* be treated as literal strings when regex=True.

Out[126]:

	Companies	Commodities	Countries	comb
0	None	None	None	"Case0":<<Companies: None. Commodities: None. ...
1	None	Recycled materials	Haiti, Honduras	"Case1":<<Companies: None. Commodities: Recycl...
2	None	Children, Child Labor, Child Abuse, Contracept...	Pakistan	"Case2":<<Companies: None. Commodities: Childr...
3	None	None	United States, Vietnam, China	"Case3":<<Companies: None. Commodities: None. ...
4	G-III Apparel Group	Apparel, Textiles, Garments, Fashion	Indonesia, China, Vietnam	"Case4":<<Companies: G-III Apparel Group. Comm...
...	...	...	...	...
903	None	None	Cambodia, Myanmar, Laos	"Case924":<<Companies: None. Commodities: None...
904	None.	None	Syria, South Sudan, Yemen, Burundi, Central Af...	"Case925":<<Companies: None.. Commodities: Non...
905	None	None	United States, United Kingdom	"Case926":<<Companies: None. Commodities: None...
906	None	None	None	"Case927":<<Companies: None. Commodities: None...
907	hershey	Cocoa, Food processing products, Agricultural ...	Côte d'Ivoire, Ghana	"Case928":<<Companies: hershey. Commodities: C...

908 rows × 4 columns

In [259]:
```
output_df[output_df.index==13]
```

Out[259]:

	Companies	Commodities	Countries	comb
13	Nike, Gap	Cotton, Textiles, Apparel, Garments	Bangladesh, Uzbekistan, India	"Case14":<<Companies: Nike, Gap.. Commodities:...

In [128]:
```
# test_country = ['Congo']
# test = output_df[output_df['Countries'].isin(test_country)]
# test
```

In [135]:
```
# Unify the country names

## DRC
output_df['Countries'] = output_df['Countries'].str.replace('Democratic Republic of Congo (DRC)', 'Democratic Republic of Congo')
output_df['Countries'] = output_df['Countries'].str.replace('Democratic Republic of the (DRC)', 'Democratic Republic of Congo')
output_df['Countries'] = output_df['Countries'].str.replace('Democratic Republic of the Congo', 'Democratic Republic of Congo')
output_df['Countries'] = output_df['Countries'].str.replace('DR Congo', 'Democratic Republic of Congo')

# Germany
output_df['Countries'] = output_df['Countries'].str.replace(' Germany', 'Germany')
# Nicaragua
output_df['Countries'] = output_df['Countries'].str.replace(' Nicaragua', 'Nicaragua')

# United States
output_df['Countries'] = output_df['Countries'].str.replace('America', 'United States')

# Brunei
output_df['Countries'] = output_df['Countries'].str.replace('Brunei', 'Brunei Darussalam')
output_df['Countries'] = output_df['Countries'].str.replace('and Brunei Darussalam', 'Brunei Darussalam')
output_df['Countries'] = output_df['Countries'].str.replace('Brunei Darussalam Darussalam', 'Brunei Darussalam')


# North Korea
output_df['Countries'] = output_df['Countries'].str.replace('North Korea', 'North Korea')

# United Kingdom
output_df['Countries'] = output_df['Countries'].str.replace('Britain', 'United Kingdom')
output_df['Countries'] = output_df['Countries'].str.replace('United Kingdom (UK)', 'United Kingdom')
output_df['Countries'] = output_df['Countries'].str.replace('the UK', 'United Kingdom')
output_df['Countries'] = output_df['Countries'].str.replace('UK', 'United Kingdom')

# Venezuela
output_df['Countries'] = output_df['Countries'].str.replace('Venezuela (Bolivarian Republic of)', 'Venezuela')

# Myanmar
output_df['Countries'] = output_df['Countries'].str.replace('Myanmar (Burma)', 'Myanmar')

# Russia
output_df['Countries'] = output_df['Countries'].str.replace('Russian Federation', 'Russia')


```

```
/var/folders/ct/fb9zr96x3g1dxcqhdytdzv5h0000gn/T/ipykernel_3558/2142745092.py:4: FutureWarning:

The default value of regex will change from True to False in a future version.

/var/folders/ct/fb9zr96x3g1dxcqhdytdzv5h0000gn/T/ipykernel_3558/2142745092.py:5: FutureWarning:

The default value of regex will change from True to False in a future version.

/var/folders/ct/fb9zr96x3g1dxcqhdytdzv5h0000gn/T/ipykernel_3558/2142745092.py:28: FutureWarning:

The default value of regex will change from True to False in a future version.

/var/folders/ct/fb9zr96x3g1dxcqhdytdzv5h0000gn/T/ipykernel_3558/2142745092.py:33: FutureWarning:

The default value of regex will change from True to False in a future version.

/var/folders/ct/fb9zr96x3g1dxcqhdytdzv5h0000gn/T/ipykernel_3558/2142745092.py:36: FutureWarning:

The default value of regex will change from True to False in a future version.
```

In [136]:
```
output_df['Countries'].value_counts()
```

Out[136]:
```
None                                                                                                                                  166
United States                                                                                                                          65
Pakistan                                                                                                                               18
India                                                                                                                                  17
Democratic Republic of Congo                                                                                                           13
                                                                                                                                     ...
Turkey, Syria,Germany, United Kingdom                                                                                                   1
Qatar, United States, Thailand, Philippines, Brazil, Canada, United Kingdom                                                             1
United States, Bosnia, Russia, Norway, France, United Kingdom, Portugal, Spain, Japan, Lithuania, Greece, Niger, Angola, Mali, Central African Republic, Somalia    1
Mexico, United States, Arizona, California, Texas, Washington, Florida, Oregon, North Carolina                                          1
United States, United Kingdom                                                                                                           1
Name: Countries, Length: 476, dtype: int64
```

In [137]:
```
country_counts= {}
for data in output_df['Countries']:
    data = data.split(", ")
    for country in data:
        if country in country_counts:
            country_counts[country] += 1
        else:
            country_counts[country] = 1

```

In [138]:
```
country_counts
```

Out[138]:
```
{'None': 199,
 'Haiti': 14,
 'Honduras': 10,
 'Pakistan': 63,
 'United States': 243,
 'Vietnam': 38,
 'China': 86,
 'Indonesia': 38,
 'Myanmar': 30,
 'Laos': 7,
 'Cambodia': 18,
 'Kyrgyzstan': 3,
 'Israel': 14,
 'India': 97,
 'Nepal': 13,
 'Bangladesh': 54,
 'Sub-Saharan Africa': 2,
 'Europe': 5,
 'Middle East': 1,
 'Afghanistan': 20
```

In [139]:
```
country_counts
```

Out[139]:
```
{'None': 199,
 'Haiti': 14,
 'Honduras': 10,
 'Pakistan': 63,
 'United States': 243,
 'Vietnam': 38,
 'China': 86,
 'Indonesia': 38,
 'Myanmar': 30,
 'Laos': 7,
 'Cambodia': 18,
 'Kyrgyzstan': 3,
 'Israel': 14,
 'India': 97,
 'Nepal': 13,
 'Bangladesh': 54,
 'Sub-Saharan Africa': 2,
 'Europe': 5,
 'Middle East': 1,
```

In [140]:
```
country_counts['Democratic Republic of Congo (DRC)']
```

Out[140]: 2

In [141]:
```
country_counts_df = pd.DataFrame(list(country_counts.items()), columns=['Country', 'Count'])
country_counts_df = country_counts_df.sort_values(by='Count', ascending=False)

# Remove "None"
country_counts_df = country_counts_df[country_counts_df['Country'] != 'None']
country_counts_df
```

Out[141]:

	Country	Count
4	United States	243
13	India	97
6	China	86
45	Syria	64
3	Pakistan	63
...	...	...
287	Ethiopia\n	1
286	Hague Conference	1
285	United Nations (UN)	1
170	Venezuela (Bolivarian Republic of)	1
371	Former Soviet States	1

371 rows × 2 columns

In [142]:
```
country_counts_df.to_csv('country_counts.csv', index=False, encoding='utf-8')
```

In [143]:
```
country_counts_df_top20 = country_counts_df[country_counts_df['Count']>=20]
country_counts_df_top20
```

Out[143]:

	Country	Count
4	United States	243
13	India	97
6	China	86
45	Syria	64
3	Pakistan	63
21	Mexico	58
15	Bangladesh	54
19	Afghanistan	39
55	Ghana	39
52	Russia	39
7	Indonesia	38
51	United Kingdom	38
5	Vietnam	38
72	Turkey	37
47	Iraq	36
23	Uzbekistan	35
34	Democratic Republic of Congo	33
8	Myanmar	30
48	Jordan	30
69	Malaysia	29
49	Lebanon	29
37	Japan	27
25	Canada	26
63	Nigeria	25
27	Thailand	25
86	Peru	21
28	Australia	21
113	Philippines	21
97	Sierra Leone	20
65	Iran	20

In [144]:
```
fig = px.choropleth(country_counts_df_top20,
                    locations='Country',
                    locationmode='country names',
                    color='Count',
                    hover_name='Country',
                    color_continuous_scale='Reds',
                    title='Frequencies of child exploitation news by country (over 20 counts only)')
fig.update_layout(width=800, height=500)
fig.write_html("country_counts.html")
fig.show()
```

5-2: Commodities

In [200]:
```
# Remove unnecessary punctuation marks.
output_df['Commodities'] = output_df['Commodities'].str.replace('\n-', ',')
output_df['Commodities'] = output_df['Commodities'].str.replace('[', '')
output_df['Commodities'] = output_df['Commodities'].str.replace(']', '')
output_df['Commodities'] = output_df['Commodities'].str.replace('.', '')
output_df['Commodities'] = output_df['Commodities'].str.replace("'", "")

# Unify commodity names
## Cobalt
output_df['Commodities'] = output_df['Commodities'].str.replace("cobolt", "cobalt")
## Apparel
output_df['Commodities'] = output_df['Commodities'].str.replace("cobolt", "cobalt")



```

```
/var/folders/ct/fb9zr96x3g1dxcqhdytdzv5h0000gn/T/ipykernel_3558/2738018453.py:3: FutureWarning:

The default value of regex will change from True to False in a future version. In addition, single character regular expressions will *not* be treated as literal strings when regex=True.

/var/folders/ct/fb9zr96x3g1dxcqhdytdzv5h0000gn/T/ipykernel_3558/2738018453.py:4: FutureWarning:

The default value of regex will change from True to False in a future version. In addition, single character regular expressions will *not* be treated as literal strings when regex=True.

/var/folders/ct/fb9zr96x3g1dxcqhdytdzv5h0000gn/T/ipykernel_3558/2738018453.py:5: FutureWarning:

The default value of regex will change from True to False in a future version. In addition, single character regular expressions will *not* be treated as literal strings when regex=True.
```

In [201]:
```
test_country = ['Cobolt']
test = output_df[output_df['Commodities'].isin(test_country)]
test
```

Out[201]:

	Companies	Commodities	Countries	comb
270	None	Cobolt	Democratic Republic of Congo	"Case279":<<Companies: None. Commodities: Cobo...
565	Samsung, Huawei, Microsoft.	Cobolt	Congo	"Case579":<<Companies: Samsung, Huawei, Micros...
667	None	Cobolt	Congo	"Case684":<<Companies: None. Commodities: ['Co...
781	GM, Renault-Nissan, Fiat Chrysler, Volkswagen,...	Cobolt	Democratic Republic of Congo (DRC)	"Case800":<<Companies: GM, Renault-Nissan, Fia...
845	Apple, HP, Samsung SDI, Sony, Dell, Foxconn, F...	Cobolt	Congo	"Case865":<<Companies: Apple, HP, Samsung SDI,...

In [202]:
```
commodity_counts= {}
for data in output_df['Commodities']:
    data = data.split(", ")
    for commodity in data:
        if commodity in commodity_counts:
            commodity_counts[commodity] += 1
        else:
            commodity_counts[commodity] = 1
```

In [203]:
```
commodity_counts
```

Out[203]:
```
{'None': 465,
 'Recycled materials': 8,
 'Children': 3,
 'Child Labor': 10,
 'Child Abuse': 2,
 'Contraceptives': 1,
 'Apparel': 115,
 'Textiles': 101,
 'Garments': 61,
 'Fashion': 1,
 'Fisheries': 2,
 'Construction': 4,
 'Agriculture': 6,
 'Books': 3,
 'Corn': 4,
 'Food Processing Products': 1,
 'Agricultural Produce': 3,
 'Child labor': 31,
 'Cotton': 37,
```

In [204]:
```
commodity_counts_df = pd.DataFrame(list(commodity_counts.items()), columns=['Commodity', 'Count'])

# Remove "None" and inappropriate words
commodity_counts_df = commodity_counts_df[~commodity_counts_df['Commodity'].isin(['None', 'Child labor','Child Labor'])]
# Reorganise
commodity_counts_df = commodity_counts_df.sort_values(by='Count', ascending=False)
commodity_counts_df.reset_index(drop=True)
```

Out[204]:

	Commodity	Count
0	Apparel	115
1	Textiles	101
2	Garments	61
3	Footwear	57
4	Agricultural produce	45
...	...	...
435	and no one makes costumes with less forced chi...	1
436	the best	1
437	can't appreciate how great are the costumes ma...	1
438	Text: "best costumes: nordstrom such a failure	1
439	Beads	1

440 rows × 2 columns

In [205]:
```
commodity_counts_df.loc[45,'Commodity']
```

Out[205]:
```
'Coffee'
```

In [206]:
```
commodity_counts_df.to_csv('commodity_counts.csv', index=False, encoding='utf-8')
```

In [207]:
```
commodity_counts_df_top20 = commodity_counts_df[:20]
commodity_counts_df_top20
```

Out[207]:

	Commodity	Count
6	Apparel	115
7	Textiles	101
8	Garments	61
27	Footwear	57
41	Agricultural produce	45
21	Electronics	44
28	Handicrafts	43
20	Gold	43
18	Cotton	37
57	Toys	37
35	Carpentry	36
19	Ores	32
33	Carpets	31
29	Furniture	30
51	Cobalt	28
30	Bricks	27
65	Cocoa	25
94	Food processing products	25
73	Cosmetics	22
66	Clothing	18

In [209]:
```

commodity_top20_freq = {commodity: count for commodity, count in zip(commodity_counts_df_top20['Commodity'], commodity_counts_df_top20['Count'])}
# WordCloudの生成
wordcloud = WordCloud(width=800, height=400, background_color='white').generate_from_frequencies(commodity_top20_freq)

# WordCloudの描画
plt.figure(figsize=(10, 5))
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis('off')
plt.title('Top 20 commodities in child exploitation news')

plt.savefig('wordcloud_commodity.jpeg', format='jpeg')
plt.show()

```

5-3: Companies

In [228]:
```
output_df['Companies'] = output_df['Companies'].str.replace('.', '')

output_df['Companies'] = output_df['Companies'].str.replace('Nestlé', 'Nestle')

output_df['Companies'] = output_df['Companies'].str.replace('adidas', 'Adidas')

output_df['Companies'] = output_df['Companies'].str.replace('Alliance One International (AOI)', 'Alliance One International')

output_df['Companies'] = output_df['Companies'].str.replace('Altria Group', 'Altria')

output_df['Companies'] = output_df['Companies'].str.replace('Apple Inc', 'Apple')
output_df['Companies'] = output_df['Companies'].str.replace('Apple Inc.', 'Apple')
output_df['Companies'] = output_df['Companies'].str.replace('Apple.', 'Apple')

output_df['Companies'] = output_df['Companies'].str.replace('Archer Daniels Midland Co', 'Archer Daniels Midland')
output_df['Companies'] = output_df['Companies'].str.replace('Archer-Daniels-Midland', 'Archer Daniels Midland')

output_df['Companies'] = output_df['Companies'].str.replace('Asos', 'ASOS')

output_df['Companies'] = output_df['Companies'].str.replace('Barry Callebaut.', 'Barry Callebaut')

output_df['Companies'] = output_df['Companies'].str.replace('bitfury group', 'bitfury')

output_df['Companies'] = output_df['Companies'].str.replace('British American Tobacco (BAT)', 'British American Tobacco')

output_df['Companies'] = output_df['Companies'].str.replace('c&a', 'C&A')
output_df['Companies'] = output_df['Companies'].str.replace('C&A Foundation', 'C&A')
output_df['Companies'] = output_df['Companies'].str.replace('C&A.', 'C&A')

output_df['Companies'] = output_df['Companies'].str.replace('Cargill Inc', 'Cargill')
output_df['Companies'] = output_df['Companies'].str.replace('Cargill.', 'Cargill')

output_df['Companies'] = output_df['Companies'].str.replace('Coca Cola.', 'Coca-Cola')
output_df['Companies'] = output_df['Companies'].str.replace('coca-cola', 'Coca-Cola')
output_df['Companies'] = output_df['Companies'].str.replace('Coca-Cola.', 'Coca-Cola')

output_df['Companies'] = output_df['Companies'].str.replace('de beers', 'De Beers')
output_df['Companies'] = output_df['Companies'].str.replace('Debeers', 'De Beers')
output_df['Companies'] = output_df['Companies'].str.replace('DeBeers', 'De Beers')
output_df['Companies'] = output_df['Companies'].str.replace('DeBeers', 'De Beers')

output_df['Companies'] = output_df['Companies'].str.replace("Driscoll", "Driscoll's")

output_df['Companies'] = output_df['Companies'].str.replace("ferrero", "Ferrero")

output_df['Companies'] = output_df['Companies'].str.replace("gap", "GAP")

output_df['Companies'] = output_df['Companies'].str.replace("h&m", "H&M")

output_df['Companies'] = output_df['Companies'].str.replace("hershey", "Hershey")
output_df['Companies'] = output_df['Companies'].str.replace("Hershey's", "Hershey")
output_df['Companies'] = output_df['Companies'].str.replace("Hershey.Âós", "Hershey")

output_df['Companies'] = output_df['Companies'].str.replace("inditex", "Inditex")

output_df['Companies'] = output_df['Companies'].str.replace("Kellogg's", "Kellogg")

output_df['Companies'] = output_df['Companies'].str.replace("Mars Incorporated", "Mars")

output_df['Companies'] = output_df['Companies'].str.replace("nike", "Nike")
output_df['Companies'] = output_df['Companies'].str.replace("Nike Inc", "Nike")

output_df['Companies'] = output_df['Companies'].str.replace("PepsiCo", "Pepsico")

output_df['Companies'] = output_df['Companies'].str.replace("samsung", "Samsung")
output_df['Companies'] = output_df['Companies'].str.replace("SAMSUNG", "Samsung")

output_df['Companies'] = output_df['Companies'].str.replace("tesco", "Tesco")

output_df['Companies'] = output_df['Companies'].str.replace("Wal-Mart", "Walmart")


```

```
/var/folders/ct/fb9zr96x3g1dxcqhdytdzv5h0000gn/T/ipykernel_3558/1626235116.py:1: FutureWarning:

The default value of regex will change from True to False in a future version. In addition, single character regular expressions will *not* be treated as literal strings when regex=True.

/var/folders/ct/fb9zr96x3g1dxcqhdytdzv5h0000gn/T/ipykernel_3558/1626235116.py:7: FutureWarning:

The default value of regex will change from True to False in a future version.

/var/folders/ct/fb9zr96x3g1dxcqhdytdzv5h0000gn/T/ipykernel_3558/1626235116.py:12: FutureWarning:

The default value of regex will change from True to False in a future version.

/var/folders/ct/fb9zr96x3g1dxcqhdytdzv5h0000gn/T/ipykernel_3558/1626235116.py:13: FutureWarning:

The default value of regex will change from True to False in a future version.

/var/folders/ct/fb9zr96x3g1dxcqhdytdzv5h0000gn/T/ipykernel_3558/1626235116.py:20: FutureWarning:

The default value of regex will change from True to False in a future version.

/var/folders/ct/fb9zr96x3g1dxcqhdytdzv5h0000gn/T/ipykernel_3558/1626235116.py:24: FutureWarning:

The default value of regex will change from True to False in a future version.

/var/folders/ct/fb9zr96x3g1dxcqhdytdzv5h0000gn/T/ipykernel_3558/1626235116.py:28: FutureWarning:

The default value of regex will change from True to False in a future version.

/var/folders/ct/fb9zr96x3g1dxcqhdytdzv5h0000gn/T/ipykernel_3558/1626235116.py:31: FutureWarning:

The default value of regex will change from True to False in a future version.

/var/folders/ct/fb9zr96x3g1dxcqhdytdzv5h0000gn/T/ipykernel_3558/1626235116.py:33: FutureWarning:

The default value of regex will change from True to False in a future version.

/var/folders/ct/fb9zr96x3g1dxcqhdytdzv5h0000gn/T/ipykernel_3558/1626235116.py:35: FutureWarning:

The default value of regex will change from True to False in a future version.
```

Type *Markdown* and LaTeX: α^2

In [233]:
```
company_counts= {}
for data in output_df['Companies']:
    data = data.split(", ")
    for company in data:
        company = company.upper()
        if company in company_counts:
            company_counts[company] += 1
        else:
            company_counts[company] = 1
company_counts
```

Out[233]:
```
{'NONE': 664,
'G-III APPAREL GROUP': 1,
'NIKE': 8,
'GAP': 10,
'DISNEY': 2,
'THAI UNION': 5,
'THAI FROZEN FOODS ASSOCIATION': 1,
'PUMA': 3,
'BELLE INTERNATIONAL': 1,
'ADIDAS': 6,
"DRISCOLL'S'S'S'S": 4,
'H&M': 14,
'INDITEX': 6,
'WALMART': 15,
'C&A': 1,
'REYNOLDS AMERICAN': 1,
'EXXONMOBIL': 2,
'ARCH COAL': 1,
'BP': 1,
'COCA-COLA': 2
```

In [234]:
```
company_counts_df = pd.DataFrame(list(company_counts.items()), columns=['Company', 'Count'])
company_counts_df = company_counts_df.sort_values(by='Count', ascending=False)

# Remove "None" and inappropriate words
company_counts_df = company_counts_df[~company_counts_df['Company'].isin(['NONE', 'None.', 'None'])]
company_counts_df
```

Out[234]:

	Company	Count
33	NESTLE	20
13	WALMART	15
11	H&M	14
31	HERSHEY	11
3	GAP	10
...	...	...
173	KAY JEWELERS	1
172	ZALES	1
171	SIGNET JEWELERS	1
170	MILLICOM	1
431	ALIF AILAAN	1

431 rows × 2 columns

In [235]:
```
company_counts_df.to_csv('company_counts.csv', index=False, encoding='utf-8')
```

In [238]:
```
company_counts_df_top20 = company_counts_df[:22]
company_counts_df_top20
```

Out[238]:

	Company	Count
33	NESTLE	20
13	WALMART	15
11	H&M	14
31	HERSHEY	11
3	GAP	10
145	SAMSUNG	9
146	APPLE	9
32	MARS	9
2	NIKE	8
63	TESCO	7
73	CARGILL	7
9	ADIDAS	6
57	SAVE THE CHILDREN	6
64	PRIMARK	6
184	FERRERO	6
12	INDITEX	6
231	FOXCONN	5
136	PEPSICO	5
5	THAI UNION	5
21	TESLA	5
70	MONDELEZ	5
181	DE BEERS	5

In [243]:

```
company_top20_freq = {company: count for company, count in zip(company_counts_df_top20['Company'], company_counts_df_top20['Count'])}
# WordCloudの生成
wordcloud = WordCloud(width=800, height=400, background_color='white').generate_from_frequencies(company_top20_freq)

# WordCloudの描画
plt.figure(figsize=(10, 5))
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis('off')
plt.title('Top 20 companies in child exploitation news')

plt.savefig('wordcloud_company.jpeg', format='jpeg')
plt.show()
```

In [ ]:

In [ ]:

In [ ]: