

```

import re
import pandas as pd
import nltk
import matplotlib.pyplot as plt
import numpy as np

from nltk.probability import FreqDist
from nltk.tokenize import word_tokenize
from cleantext import clean
from nltk.lm import Vocabulary

nltk.download('punkt')

[nltk_data] Downloading package punkt to /home/andreas-linus-thalund-
[nltk_data] midtgaard/nltk_data...
[nltk_data] Package punkt is already up-to-date!

True

#test data
#cpr
cpr_list = [
    "200144-7290", #2000
    "020164-1952", #1900
    "1303494374", #1900
    "2006681112", #1900
    "280659-4979", #1900
    "050421-8939", #2000
    "1401704600", #1900
    "2708256898", #2000
    "141100-6779", #2000
    "211183-3482", #1900
    "2507371827", #1900
    "2510439716", #1900
    "1601594544", #1900
    "0808259665", #2000
    "2102077509", #2000
    "2503102352", #1900
    "130284-0998", #1900
    "1705956426", #1800
    "180263-5407", #1800
    "0604882964", #1900
]

#matches the cpr numbers above by index to a century in accordance
with the table in the problemstatement
CPR_years = [
    "2000",
    "1900",
    "1900",
    "1900",

```

```

    "1900",
    "2000",
    "1900",
    "2000",
    "2000",
    "1900",
    "1900",
    "1900",
    "1900",
    "1900",
    "2000",
    "2000",
    "1900",
    "1900",
    "1800",
    "1800",
    "1900",
]

invalid_cpr_list = [
    "1234567890",      # Missing separator but invalid format (should
    be DDMMYYIIII)
    "123456-789",      # Too short (only 3 digits in the last part)
    "12345-67890",      # Too many digits in the first part
    "1234-567890",      # Incorrect separator placement
    "123456_7890",      # Wrong separator (underscore instead of
    hyphen)
    "12A456-7890",      # Contains a letter in the date part
    "123456-78B0",      # Contains a letter in the identifier part
    "000000-0000",      # All zeros (not realistic)
    "320299-1234",      # Invalid date (February 32nd does not exist)
    "311199-12345",      # Too many digits in the last part
    "311299-",          # Missing last 4 digits
    "-3112991234",      # Starts with a separator
    "311299--1234",      # Double separator
    "999999-9999",      # Unrealistic high values
    "010120 1234",      # Space instead of hyphen
    "010120123",        # Last part too short
    "abcdef-ghij",      # Only letters
    "01.01.20-1234",    # Wrong date format (dots instead of numbers)
    "310499/5678",      # Wrong separator (/ instead of -)
]

#url
# URLs that should match
valid_urls = [
    "http://example.com",
    "https://example.com",

```

```

"http://www.example.com",
"https://sub.example.com",
"http://example.com:8080",
"https://example.com/path/to/page",
"http://example.com?query=param",
"https://example.com#fragment",
"http://user:pass@example.com",
"https://user:pass@sub.example.com",
"http://192.168.1.1",
"https://255.255.255.255",
"http://example.com:3000/path?query=test#frag",
"https://sub.example.co.uk",
"http://example.com/index.html",
"https://example.com/login?user=test&pass=123",
"https://en.wikipedia.org/wiki/URL",
"https://absalon.ku.dk/courses/80486/assignments/232056",
"https://docs.python.org/3/library/re.html",
"https://www.minkompetencemappe.dk/"
"https://people-vol.roskilde-festival.dk/dashboard/"

"https://dans.stads.dk/SelfUserRegistration/faces/WelcomePage.jspx;jse
ssionid=N3oGQvXwtmcGNIPKLHLSmdJ_457FCGtLNein7bSP1m9U67Ky-SnT!
921246790"
    "https://www.overleaf.com/project"
    "https://id.ku.dk/nidp/wsfed/ep?id=preauth-ku-secure-
kerberos&sid=0&option=credential&sid=0"

]

# URLs that should NOT match
invalid_urls = [
    "htt://example.com",          # Missing 'p' in http
    "http://example.com",        # Missing ':'
    "http:/example.com",         # Missing second '/'
    "://example.com",            # Missing scheme
    "http://",                   # No domain
    "https://.com",              # Invalid domain
    "http://example.com:-80",    # Invalid port number
    "http://example.com/ space", # Space in path
    "http://example.com?query= space", # Space in query
    "http://user:pa:ss@example.com", # Invalid user info
    "http://example.com#frag ment", # Space in fragment
]

valid_emails = [
    "john.doe@gmail.com",
    "jane_doe123@hotmail.com",
    "user+test@gmail.net",

```

```

    "my.email-123@hotmail.org",
    "example_456@gmail.com",
    "hello.world+test@hotmail.com",
    "valid_email123@gmail.org",
    "user.name1234@hotmail.net",
    "name.surname-xyz@gmail.net",
    "contact_me+info@hotmail.org"
]

invalid_emails = [
    "missingatgmail.com", # Missing @
    "wrongformat@gmail.com", # Double @
    "user@invalid_domain.xyz", # Invalid domain
    "@gmail.com", # Missing username
    "no_domain@.com", # Missing domain
    "plainaddress", # No @ symbol
    "user.name@gmailcom" # Missing dot in domain
]

```

PART 1

```

#cpr regex som matcher ddmmyyIIII og ddmmyy-IIII in four groups DD,
MM, YY, and IIII
cprRE = r"^(0[1-9]|[1-2][0-9]|3[0-1])(0[1-9]|1[0-2])([0-9]{2})-?([0-9]{4})$"

#function finding century of birth based on cpr
def yearofbirth (input_IIII, input_YY):
    #Define IIII/YY&century relationship
    table = [
        ((1, 3999), (0, 99), 1900),
        ((4000, 4999), (0, 36), 2000),
        ((4000, 4999), (37, 99), 1900),
        ((5000, 8999), (0, 57), 2000),
        ((5000, 8999), (58, 99), 1800),
        ((9000, 9999), (0, 36), 2000),
        ((9000, 9999), (37, 99), 1900),
    ]

    #check input and return century in rows if both IIII and YY
    tuples match
    for (range_IIII, range_YY, century) in table:
        if int(range_IIII[0]) <= int(input_IIII) <= int(range_IIII[1])
        and int(range_YY[0]) <= int(input_YY) <= int(range_YY[1]):
            return century

    #else raise error
    raise ValueError("Invalid input")

```


Correct
Correct

PART 2

1. Download CSV

Downloaded from: https://raw.githubusercontent.com/several27/FakeNewsCorpus/master/news_sample.csv

2. Read the CSV file into memory

```
#load data
dataPath = "data/"
newsSample = pd.read_csv(dataPath + "news_sample.csv")
nsdf_raw = pd.DataFrame(newsSample)
nsdf = pd.DataFrame(newsSample)
nsdf = nsdf.reset_index(drop=True) # Reset index??
```

3. Data Inspection

- Index column was wierd
- Missing all keywords and summary
- Missing most of tags and Meta_discription. remove or If deemed usable do Imputation of missing data
- meta_description looks full but is mostly empty lists. remove or If deemed at all usable do Imputation of missing data or at least convert these empty lists to nan or similar
- If used for binary classification like in fakenews project, the labels in the type column should be grouped.
- In the content column i have seen different Currencies and methods for writting proper nouns (Entity names) so for effective NLP these need to be unified

```
print(nsdf.info()) # Check column types and missing values
print(nsdf['keywords'])
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 250 entries, 0 to 249
Data columns (total 16 columns):
#   Column                Non-Null Count  Dtype
---  -
0   Unnamed: 0            250 non-null   int64
1   id                    250 non-null   int64
2   domain                250 non-null   object
3   type                  238 non-null   object
4   url                   250 non-null   object
5   content               250 non-null   object
6   scraped_at            250 non-null   object
7   inserted_at           250 non-null   object
```

```

8   updated_at      250 non-null    object
9   title           250 non-null    object
10  authors          170 non-null    object
11  keywords         0 non-null     float64
12  meta_keywords    250 non-null    object
13  meta_description 54 non-null     object
14  tags             27 non-null     object
15  summary          0 non-null     float64
dtypes: float64(2), int64(2), object(12)
memory usage: 31.4+ KB
None
0     NaN
1     NaN
2     NaN
3     NaN
4     NaN
..
245   NaN
246   NaN
247   NaN
248   NaN
249   NaN
Name: keywords, Length: 250, dtype: float64

```

4. Cleaning

My own clean_text() function

```

def clean_text(data, column):
    def cleantext_manual_help(text):
        if isinstance(text, str):
            # Convert text to lowercase
            text = text.lower()

            # Replace numbers, dates, emails, and URLs
            text = re.sub(r"\d+", "<NUM>", text) # Replace numbers
            text = re.sub(r"(0[1-9]|[1-2][0-9]|3[0-1])[-./]?([0-9]|1[0-2])[-./]?([0-9]{2}|[0-9]{4})", "<DATE>", text) # Replace date type 1
            text = re.sub(r"(0[1-9]|[1-2][0-9]|3[0-1])\s([A-Za-z]{3})\s([0-9]{2}|[0-9]{4})", "<DATE>", text) # Replace date type 2
            text = re.sub(r"[a-zA-Z0-9._+-]+@(gmail|hotmail).(net|com|org)", "<EMAIL>", text) # Replace emails
            text = re.sub(r"^(https?):\/\/(?:([a-zA-Z0-9._%+-]+):([a-zA-Z0-9._%+-]+)@)?([a-zA-Z0-9.-]+\.[a-zA-Z]{2,}|(?:\d{1,3}\.){3}\d{1,3})(:\d{1,5})?(\/[^\s#]*)?(\/[^\s#]*)?(#[^\s]*)?$", "<URL>", text) # Replace URLs - url syntax form:
            https://en.wikipedia.org/wiki/URL

```

```

        # Remove excess whitespace
        text = re.sub(r"\s+", " ", text).strip()

        return text

    return text # Return unchanged if not a string
data[column] = data[column].apply(clean_text_manual_help) # Apply
function
return data

# userinfo: (?:(?:[a-zA-Z0-9._%+-]+):(?:[a-zA-Z0-9._%+-]+)@)?
# host: ([a-zA-Z0-9.-]+\.[a-zA-Z]{2,}|(?:\d{1,3}\.){3}\d{1,3})
# fragment: (#[^\s]*)?

nsdf_cleaned = clean_text(nsdf, 'content')
print(nsdf_cleaned['content'].head)

<bound method NDFrame.head of 0      sometimes the power of christmas
will make you...
1      awakening of <NUM> strands of dna – “reconnect...
2      never hike alone: a friday the <NUM>th fan fil...
3      when a rare shark was caught, scientists were ...
4      donald trump has the unnerving ability to abil...
...
245     prison for rahm, god’s work and many others he...
246     <NUM> useful items for your tiny home headline...
247     former cia director michael hayden said thursd...
248     antonio sabato jr. says hollywood's liberal el...
249     former u.s. president bill clinton on monday c...
Name: content, Length: 250, dtype: object>

#test af cleantext regexes
print("failed url valid:")
urlRE = r"^(https?):\/\/(?:([a-zA-Z0-9._%+-]+):([a-zA-Z0-9._%+-]+)@)?
([a-zA-Z0-9.-]+\.[a-zA-Z]{2,}|(?:\d{1,3}\.){3}\d{1,3})(?:\d{1,5})?(\/
[^\s?#]*)?(\/[^\s#]*)?(#[^\s]*)?$" #https://en.wikipedia.org/wiki/URL
for url in valid_urls:
    match = re.match(urlRE, url)
    if not match:
        print(url)

print("failed url invalid:")

for url in invalid_urls:
    match = re.match(urlRE, url)
    if match:

```



```

        print(url)

print("failed email valid:")
urlRE = r"[a-zA-Z0-9._-]+@(gmail|hotmail).(net|com|org)"
for email in valid_emails:
    match = re.match(urlRE, email)
    if not match:
        print(url)

print("failed email invalid:")

for email in invalid_emails:
    match = re.match(urlRE, email)
    if match:
        print(email)

#date and num tested by tjecking matches in data manually

failed url valid:
failed url invalid:
failed email valid:
failed email invalid:

```

5. Cleaning Using Library

```

def clean_text_help(text):
    if isinstance(text, str):
        #replace dates
        text = re.sub(r"(0[1-9]|[1-2][0-9]|3[0-1])[-./]?([0-9]|1[0-2])[-./]?([0-9]{2}|[0-9]{4})", "<DATE>", text) # Replace date type 1
        text = re.sub(r"(0[1-9]|[1-2][0-9]|3[0-1])\s([A-Za-z]{3})\s([0-9]{2}|[0-9]{4})", "<DATE>", text) # Replace date type 2
        return clean(text, lower=True, fix_unicode=True,
            replace_with_url="<URL>", replace_with_email="<EMAIL>",
            replace_with_number="<NUM>", replace_with_digit="<NUM>")
    return text # Return unchanged if not a string

def cleanText(data, column):
    data[column] = data[column].apply(clean_text_help) # Apply
function
    return data

nsdf_cleaned = cleanText(nsdf, 'content')
print(nsdf_cleaned['content'].head)

<bound method NDFrame.head of 0      sometimes the power of christmas
will make you...
1      awakening of <num> strands of dna "reconnectin...

```

```

2      never hike alone: a friday the <num>th fan fil...
3      when a rare shark was caught, scientists were ...
4      donald trump has the unnerving ability to abil...
...
245    prison for rahm, god's work and many others he...
246    <num> useful items for your tiny home headline...
247    former cia director michael hayden said thursd...
248    antonio sabato jr. says hollywood's liberal el...
249    former u.s. president bill clinton on monday c...
Name: content, Length: 250, dtype: object>

```

PART 3

1. Number of unique words

```

#making frequency dictionary
def getFrequency(data, column):
    N = data.shape[0] # Get the number of rows
    allWords = []
    for text in data[column]:
        if isinstance(text, str):
            allWords.extend(word_tokenize(text))
    return FreqDist(allWords)

#word frequency pre preprocessing
print("word frequency pre preprocessing")
word_frequency_pre = getFrequency(nsdf_raw, 'content')
print(sum(word_frequency_pre.values()))

word frequency pre preprocessing
198366

#word frequency post preprocessing
print("word frequency post preprocessing")
word_frequency_post = getFrequency(nsdf_cleaned, 'content')
print(sum(word_frequency_post.values()))

word frequency post preprocessing
204146

pre = sum(word_frequency_pre.values())
post = sum(word_frequency_post.values())
print("Difference")
print(abs(pre - post))

Difference
5780

```

1. Word Frequency

```

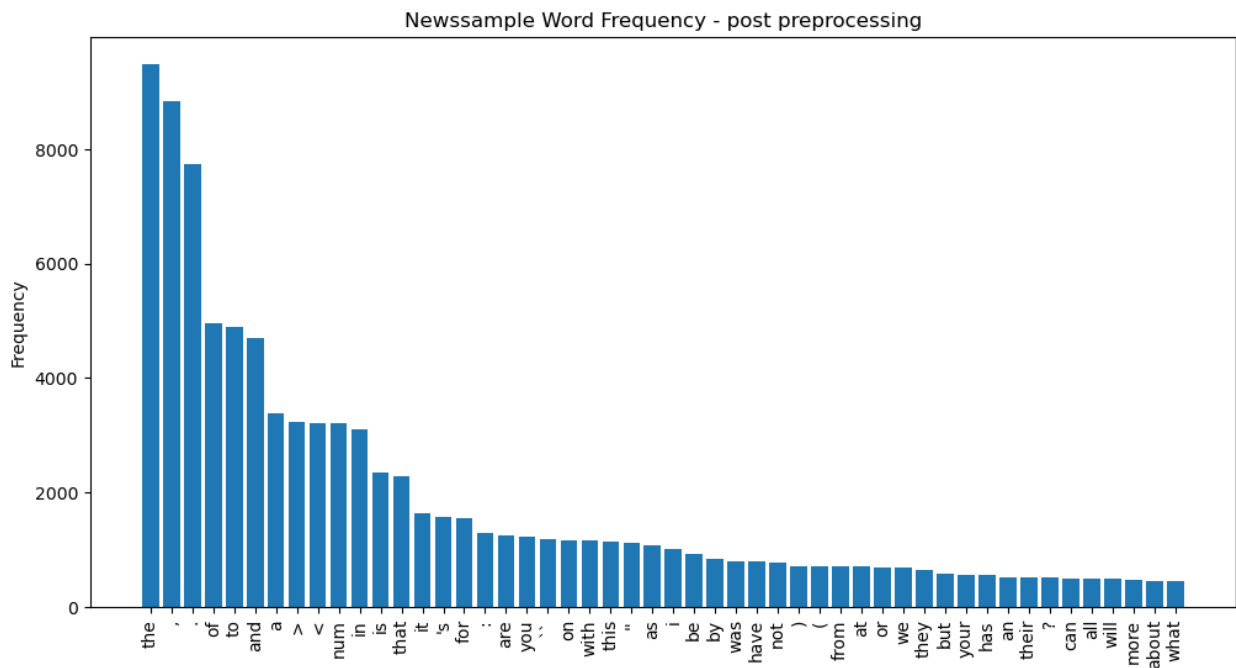
fig, ax = plt.subplots(figsize=(12, 6))
frequency = word_frequency_post.most_common(50)
words, counts = zip(*frequency)

ax.bar(words, counts, color= 'tab:blue')

ax.set_ylabel('Frequency')
ax.set_title('Newssample Word Frequency - post preprocessing')
plt.xticks(rotation=90)

plt.show()

```



```

fig, ax = plt.subplots(figsize=(12, 6))
frequency = word_frequency_pre.most_common(50)
words, counts = zip(*frequency)

ax.bar(words, counts, color= 'tab:green')

ax.set_ylabel('Frequency')
ax.set_title('Newssample Word Frequency - pre preprocessing')
plt.xticks(rotation=90)

plt.show()

```

Newssample Word Frequency - pre preprocessing

