IMPORTING RELEVANT INITIAL LIBRARIES

In [1]:
```
import requests
from bs4 import BeautifulSoup
from nltk.tokenize import word_tokenize
from nltk.corpus import words
```

In [2]:
```
import nltk
# Download the 'words' dataset from NLTK
nltk.download('words')
```

```
[nltk_data] Downloading package words to
[nltk_data]     /Users/nishandhillon/nltk_data...
[nltk_data]   Package words is already up-to-date!
```

Out[2]: True

FUNCTION TO FILTER URLS WITH RELEVANT KEYWORDS

In [3]:
```
import re

# Given keywords and compiled patterns
keywords = ['empire', 'kingdom']
keyword_patterns = [re.compile(keyword, re.IGNORECASE) for keyword in
# Function to check if any keyword matches part of the URL
def check_keywords_in_url(url, patterns):
    for pattern in patterns:
        if pattern.search(url):
            return True  # Return True if any keyword matches
    return False  # Return False if no keyword matches
```

LIST OF URL'S TO SCRAP THROUGH

In [4]:
```
URLS= ['https://history.howstuffworks.com/world-history/10-long-lived-
       'https://www.businessinsider.com/the-10-greatest-empires-in-his
```

SCRAPPING THROUGH THE INITIAL URL'S TO GATHERE MORE RELEVANT URL'S

In [5]:
```
total_links = set()
for url in URLS:
    page = requests.get(url)
    soup = BeautifulSoup(page.text, 'html.parser')
    links = soup.find_all('a')
    external_urls = [link.get('href') for link in links if link.get('h
    for external_url in external_urls:
        if check_keywords_in_url(external_url, keyword_patterns):
            if external_url not in total_links:
                total_links.add(external_url)
            if len(total_links) > 100:
                break
```

In [6]:
```
len(total_links)
```

Out[6]:
```
34
```

In [7]:
```
print(total_links)
```

```
{'http://en.wikipedia.org/wiki/Mongol_empire', 'https://portugal.co
m/portugal-blogs/the-portuguese-empire', 'https://insider-app.onelin
k.me/4cpG/?af_js_web=true&af_ss_ver=2_3_0&af_dp=insider%3A%2F%2Fbi%2
Fpost%2Fthe-10-greatest-empires-in-history-2011-9&af_force_deeplink=
true&is_retargeting=true&deep_link_value=https%3A%2F%2Fwww.businessi
nsider.com%2Fthe-10-greatest-empires-in-history-2011-9&pid=businessi
nsider&c=post_page_share_bar_v2_smart_4.13.23', 'https://www.worldat
las.com/geography/mongol-empire.html', 'http://en.wikipedia.org/wik
i/Russian_empire', 'https://bestdiplomats.org/leaders-of-ottoman-emp
ire/', 'https://www.worldatlas.com/geography/russian-empire.html',
'https://www.worldatlas.com/maps/united-kingdom', 'https://www.newwo
rldencyclopedia.org/entry/Kanem-Bornu_Kingdom', 'https://www.worldat
las.com/history/the-fall-of-the-russian-empire.html', 'http://www.bb
c.co.uk/religion/religions/islam/history/ottomanempire_1.shtml', 'ht
tps://history.howstuffworks.com/world-history/ottoman-empire.htm',
'http://www.britannica.com/EBchecked/topic/102315/history-of-Central
-Asia/73543/Creation-of-the-Mongol-empire', 'https://www.worldatlas.
com/articles/genghis-khan-of-the-mongol-empire-world-leaders-in-hist
ory.html', 'https://www.worldhistory.org/Khmer_Empire/', 'https://hi
storyradio.org/2018/01/05/zhou-daguan-and-his-12th-century-journey-t
o-an-empire-lost-in-time/', 'http://en.wikipedia.org/wiki/British_em
pire', 'http://en.wikipedia.org/wiki/Portuguese_empire', 'https://ww
w.worldatlas.com/geography/10-most-long-lived-empires-in-history.htm
l', 'https://www.worldhistory.org/empire/', 'https://ehistory.osu.ed
u/articles/ottoman-empire', 'https://www.worldatlas.com/articles/wha
t-does-the-sun-never-sets-on-the-british-empire-mean.html', 'http
s://www.worldhistory.org/Kingdom_of_Kanem/', 'http://en.wikipedia.or
g/wiki/List_of_largest_empires', 'https://www.newworldencyclopedia.o
rg/entry/Holy_Roman_Empire', 'https://www.worldatlas.com/geography/b
ritish-empire.html', 'https://teams.microsoft.com/share?href=http
s://education.nationalgeographic.org/resource/mauryan-empire/', 'htt
ps://www.worldhistory.org/Roman_Empire/', 'https://classroom.google.
com/share?url=https://education.nationalgeographic.org/resource/maur
yan-empire/', 'https://www.newworldencyclopedia.org/entry/Ethiopian_
Empire', 'https://www.worldatlas.com/geography/second-french-colonia
l-empire.html', 'https://bestdiplomats.org/largest-empires-in-histor
y/#respond', 'https://bestdiplomats.org/why-roman-empire-fell/', 'ht
tps://bestdiplomats.org/tag/largest-empires-in-history/'}
```

SCRAPPING TEXT OFF EACH PAGE AND WRITING IT TO TEXT FILE

In [8]:
```
i = 1
for link in total_links:
    page = requests.get(link)
    soup = BeautifulSoup(page.content, 'html.parser')
    text = soup.get_text()
    filename = 'file' + str(i) + '.txt'
    f = open(filename, 'w')
    f.write(text)
    f.close()
    i += 1
```

CLEANING UP THE FILES AND SAVING THE CLEANED UP TEXT TO AN OUTPUT FILE

In [9]:
```
import re
from nltk.corpus import stopwords
from nltk.stem import PorterStemmer
from nltk.stem import WordNetLemmatizer
from nltk.tokenize import sent_tokenize

stop_words = set(stopwords.words('english'))
stemmer = PorterStemmer()
lemmatizer = WordNetLemmatizer()

# Load set of English words
english_words = set(words.words())

# CLEANING UP THE RAW FILES AND SAVING THEM AS OUTPUT FILES
for i in range(1, len(total_links) + 1):
    filename = 'file' + str(i) + '.txt'
    output_filename = 'output' + str(i) + '.txt'
    f_output = open(output_filename, 'w')
    with open(filename,'r') as f:
        lines = f.read().splitlines()
    for line in lines:
        # line = re.sub(r'[.?!,:;()\-\n\d]',' ', line.lower())
        tokens = word_tokenize(line)
        # Removing stopwords and stemming
        cleaned_tokens = [lemmatizer.lemmatize(word) for word in toker
        # removing words which are not in english
        filtered_tokens = ' '.join([word for word in cleaned_tokens if
        sentences = sent_tokenize(filtered_tokens)
        for sentence in sentences:
            f_output.write(sentence + '\n')
    f_output.close()
```

CREATING A CORPUS FROM CLEANED UP OUTPUT FILES

In [10]:
```
corpus = []
```

In [11]:
```
# CREATING CORPUS FROM THE CLEANED UP FILES
corpus = []
import re
for i in range(1, 35):
    filename = 'output' + str(i) + '.txt'
    words_to_remove = ['\t', 'th']
    with open(filename, 'r') as f:
        content = f.read().lower()
        pattern = '[^\w\s]'
        content = re.sub(pattern, '', content)
        for word in words_to_remove:
            content = content.replace(word, '')
        corpus.append(content)
```

CREATNG KNOWLEDGE BASE AFTER APPLYING TF-IDF AND OTHER TECHNIQUES

In [12]:
```
from sklearn.feature_extraction.text import TfidfVectorizer
import numpy as np



# Words to remove from the final list
words_to_remove = ['new', 'people', 'share', 'united', 'university', '
                   , 'china','known','area', 'million', 'led', 'east',
                   'moment', 'york', 'article', 'cooky', 'khanate', 'de

# Initialize TF-IDF Vectorizer
vectorizer = TfidfVectorizer(stop_words='english')

# Fit and transform the corpus
X = vectorizer.fit_transform(corpus)

# Sum TF-IDF scores for each term across all documents
sum_tfidf = np.array(X.sum(axis=0)).flatten()

# Get the feature names (words/terms)
words = np.array(vectorizer.get_feature_names())

# Sort the scores
sorted_indices = np.argsort(sum_tfidf)[::-1]

# Initialize an empty list to hold the top words excluding those to be
filtered_top_words = []

# Iterate over sorted indices and add words to the filtered list if th
for index in sorted_indices:
    if words[index] not in words_to_remove:
        filtered_top_words.append(words[index])
    # Stop once we have the top 25 words after filtering
    if len(filtered_top_words) == 40:
        break

# Display the filtered top 25 words
print(filtered_top_words)
```

```
['empire', 'history', 'war', 'century', 'kingdom', 'dynasty', 'colon
y', 'advertisement', 'territory', 'king', 'power', 'emperor', 'milit
ary', 'trade', 'policy', 'press', 'rule', 'geography', 'original',
'political', 'imperial', 'government', 'sultan', 'second', 'sea', 'w
est', 'control', 'end', 'city', 'home', 'country', 'science', 'reig
n', 'search', 'independence', 'contact', 'central', 'cultural', 'mar
ch', 'sign']
```

BASED ON THE TOP 25 WORDS AND PRIOR KNOLEDGE BASE, CREATING KNOWLEDGE BASE

In [13]:
```
knowledge_base = [
    'Empire: A sovereign state comprising multiple territories and peo
    'History: The study of past events, particularly significant polit
    'Century: A period of 100 years, often used as a milestone to deli
    'War: A state of armed conflict between different nations or state
    'Colony: A territory under the immediate political control of a st
    'Emperor: The ruler of an empire, commanding vast territories and
    'Dynasty: A line of hereditary rulers of an empire or kingdom, oft
    'Trade: The exchange of goods and services, crucial for the econom
    'Military: The armed forces of an empire, instrumental in defense,
    'Government: The system by which a state or community is governed,
    'Cultural: Pertaining to the arts, customs, traditions, and achiev
    'City: A large and significant settlement, often serving as admini
    'Imperial: Relating to an empire or emperor, denoting authority, c
    'Independence: The condition of a nation, country, or state which
    'Sultan: A title used in Muslim countries for a ruler or nobleman,
]
```

PICKLING KNOWLEDGE BASE

In [14]:
```
import pickle
pickle.dump(knowledge_base, open('kb.p', 'wb'))
```