```
# Imports and downloads
import nltk
nltk.download('punkt')
nltk.download('stopwords')
nltk.download('wordnet')
from collections import Counter
from sklearn.feature_extraction.text import TfidfVectorizer
import requests
from bs4 import BeautifulSoup
from urllib.parse import urlparse, urljoin
from urllib import request
import string
import os
import pickle
import re
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package wordnet to /root/nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
```

```
'''
Web Crawler function to create a knowledge base using a custom web crawler.
Takes the starting url, the maximum links, and the maximum depth as parameters.
'''
def web_crawler(start_url, max_links, max_depth):

    #Keeping track of crawled urls and unique domains
    urls_visited = set()
    unique_domains = set()

    #Keeping track of relevant urls
    urls_relevant = []

    #Creating a queue of URLs to crawl
    queue = [(start_url, 0)]

    while queue and len(urls_relevant) < max_links:
        url, num = queue.pop(0)

        if url not in urls_visited and num <= max_depth:
            try:
                #Getting text from URL
                html = requests.get(url).text

                #Parsing the HTML content
                soup = BeautifulSoup(html, 'html.parser')

                #Storing the text in a file
                filename = f'{urlparse(url).hostname}.txt'
                with open(filename, 'w') as file:
                    for p in soup.find_all('p'):
                        file.write(p.get_text() + '\n')

                #Finding links in the page
                if num < max_depth:
                    for link in soup.find_all('a', href = True):
                      absolute_link = urljoin(url, link['href'])
                      domain = urlparse(absolute_link).hostname
                      if absolute_link not in urls_visited and domain not in unique_domains:
                        queue.append((absolute_link, num + 1))
                        unique_domains.add(domain)

                urls_relevant.append(url)
                urls_visited.add(url)
                print(f"Crawled: {url}")

            except requests.RequestException as e:
                print(f"Failed to get {url}")

    return urls_relevant[:max_links]

# Call the web_crawler function
start_url = 'https://www.biography.com/actors/leonardo-dicaprio'
```

```
urls_relevant = web_crawler(start_url, 25, 2)

     Crawled: https://www.biography.com/actors/leonardo-dicaprio
     Crawled: https://www.biography.com/search
     Crawled: https://www.hearst.com/-/us-magazines-privacy-notice
     Crawled: https://go.redirectingat.com/?id=74968X1712615&url=https%3A%2F%2Fwww.fandango.com%2Fkillers-of-the-flower-moon-2023
     Crawled: https://www.indiewire.com/features/general/flower-moon-script-changes-jesse-plemons-dicaprio-role-1234617674/
     Crawled: https://www.npr.org/2014/01/10/261081863/a-wolf-on-the-loose-and-loving-the-carnage
     Crawled: https://www.backstage.com/magazine/article/leonardo-dicaprio-embodying-j-edgar-hoover-55217/
     Crawled: https://www.thelist.com/779841/why-leonardo-dicaprio-was-advised-to-change-his-name-early-in-his-career/
     Crawled: https://www.imdb.com/event/ev0000716/1991/1/
     Crawled: https://www.rogerebert.com/reviews/this-boys-life-1993
     Crawled: https://www.nytimes.com/1993/12/12/movies/up-coming-leonardo-dicaprio-actor-boyishly-handsome-that-s-liability.html
     Crawled: https://www.amazon.com/dp/B007XDXRR0
     Crawled: https://www.rottentomatoes.com/m/aviator/reviews?type=top_critics
     Failed to get https://collider.com/craziest-method-acting-for-famous-roles/
     Crawled: https://content.time.com/time/arts/article/0,8599,1562640-2,00.html
     Crawled: http://www.cnn.com/2009/SHOWBIZ/Movies/01/23/kate.leo/index.html
     Crawled: https://variety.com/2014/film/news/jonah-hill-was-paid-60000-for-wolf-of-wall-street-1201066745/
     Crawled: https://www.theguardian.com/film/2013/dec/31/leonardo-dicaprio-defends-wolf-of-wall-street
     Crawled: https://aaspeechesdb.oscars.org/link/088-1/
     Crawled: https://canoe.com/entertainment/movies/leonardo-dicaprio-took-pay-cut-for-once-upon-a-time-in-hollywood-role
     Crawled: https://www.latimes.com/entertainment-arts/movies/story/2023-10-19/killers-of-the-flower-moon-review-martin-scorses
     Crawled: https://www.celebritynetworth.com/richest-celebrities/actors/leonardo-dicaprio-net-worth/
     Crawled: https://twitter.com/biography
     Crawled: https://www.facebook.com/Biography
     Crawled: https://www.instagram.com/biography/
     Crawled: http://subscribe.hearstmags.com/circulation/shared/index.html


'''
Cleaned files function to clean up the text files.
Takes the list of files as parameter
'''
def cleaned_files(file_list):
  cleaned_files = []

  for file in file_list:
    with open(file, 'r') as f:
      text = f.read()

    #Removing newlines from the text
    text = text.replace('\n', '')

    #Lowercase the text
    text = text.lower()

    #Tokenizing the text
    tokens = nltk.word_tokenize(text)

    #Removing stopwords and punctuation
    stop_words = set(nltk.corpus.stopwords.words('english'))
    new_tokens = [token for token in tokens if token.isalpha() and token not in stop_words]

    #Get the new cleaned text
    cleaned_text = '\n'.join(new_tokens)

    #Write sentences to a new file
    with open(f'cleaned_{file}', 'w') as output:
      output.write(cleaned_text)

    cleaned_files.append(cleaned_text)

  return cleaned_files

# Call the cleaned_files function
directory_files = [file for file in os.listdir() if file.endswith(".txt")]
cleaned_files_output = cleaned_files(directory_files)
```

```
'''
Important words function to get 40 important terms from the cleaned-up files.
Takes the list of cleaned files and the number of terms needed as a parameter.
'''
def important_words(cleaned_files, num_terms):

    #Geting the text from all files and combining it into one
    all_text = ' '.join(cleaned_files)

    #Tokenizeing the text
    tokens = nltk.word_tokenize(all_text)

    #Lowercase and remove stop words and punctuation
    stop_words = set(nltk.corpus.stopwords.words('english'))
    tokens = [token.lower() for token in tokens if token.isalpha() and token not in stop_words]

    #Lemmatizing the words
    lemmatizer = nltk.WordNetLemmatizer()
    terms = [lemmatizer.lemmatize(token) for token in tokens]

    #Create a tf-idf vectorizer
    tfidf_vectorizer = TfidfVectorizer()
    tfidf_matrix = tfidf_vectorizer.fit_transform([all_text])

    #Getting the feature names
    feature_names = tfidf_vectorizer.get_feature_names_out()

    # Get TF-IDF scores for each term
    tfidf_scores = tfidf_matrix.toarray()[0]

    #Creating terms dictionary and the tf-idf scores
    terms_dict = dict(zip(feature_names, tfidf_scores))

    #Sorting the terms based on descending order
    sorted_terms = sorted(terms_dict.items(), key=lambda x: x[1], reverse=True)

    #Getting the top important terms
    top_terms = [term[0] for term in sorted_terms[:num_terms]]

    return top_terms

# Calling the important_words function
important_terms = important_words(cleaned_files_output, 40)
print("Important terms determined by function: ")
print(important_terms)
```

```
Important terms determined by function:
['dicaprio', 'people', 'film', 'movie', 'million', 'leonardo', 'one', 'going', 'time', 'diamonds', 'certainly', 'industry',
```

```
# Function to get relevant sentences containing a the specific term from the text
def get_facts(text, term):
    # Tokenizing the text into sentences
    sentences = nltk.sent_tokenize(text)

    #Getting the sentences that contain the term
    relevant_sent = [sentence.strip() for sentence in sentences if term.lower() in sentence.lower()]

    return relevant_sent

#Function to update the knowledge base with facts from a provied URL
def update_knowledge_base(url, terms, knowledge_base):

    #Getting the html content from the url
    html = requests.get(url).text
    soup = BeautifulSoup(html, 'html.parser')

    #Get the content
    content = " ".join([paragraph.get_text() for paragraph in soup.find_all('p')])

    #Iterating through the given terms
    for i in terms:
      #Getting the relevant sentences for each term
      relevant_sent = get_facts(content, i)

      #Adding only one fact about the term to the knowledge base
      if relevant_sent:
        knowledge_base[i.lower()] = relevant_sent[0]
```

```
# Initializing the knowledge base dictionary
knowledge_base = {}

#Important terms that are relevant from the important_words() function
terms = ['dicaprio','film','one','movie','leonardo','million','time','actor','part','issue','diamond','issues','scorsese','hollyw

#Automarically adding the facts to the knowledge base
for url in urls_relevant:
    update_knowledge_base(url, terms, knowledge_base)

# Manually adding additional terms and facts to the knowledge base
knowledge_base['age'] = 'Leonardo DiCaprio is 49 years old.'
knowledge_base['height'] = 'Leonardo DiCaprio is approximately 6 feet tall.'
knowledge_base['producer'] = 'In 2013, Leonardo DiCaprio collaborated with  Martin Scorsese to star in and co-produce The Wolf of
knowledge_base['oscar'] = 'Leonardo DiCaprio received the Oscar for Best Actor for the 2015 film, The Revenant.'
knowledge_base['awards'] = 'Leonardo DiCaprio\'s iconic film, Titanic, achieved immense success both critically and commercially.
knowledge_base['titanic'] = 'Leonardo DiCaprio\'s Titanic was the first film to reach the billion dollar mark in international sa
knowledge_base['star'] = 'Leonardo DiCaprio has starred in  Quentin Tarantino\'s works such as Django Unchained and Once Upon a T
knowledge_base['perform'] = 'In preparation for his role in the 1993 film What’s Eating Gilbert Grape?, Leonardo DiCaprio spent s
knowledge_base['born'] = 'Leonardo DiCaprio was born on November 11, 1974, in Los Angeles, California, USA.'
knowledge_base['movies'] = 'Leonardo DiCaprio has appeared in several well-known films, such as Titanic, Inception, The Revenant,

#Saving the knowledge base as a pickle file
with open('knowledge_base.pkl', 'wb') as file:
    pickle.dump(knowledge_base, file)

#Printing the knowledge base
for term, fact in knowledge_base.items():
    print(f"{term.capitalize()}: {fact}")
```

```
Dicaprio: Leonardo DiCaprio is an American actor, producer, philanthropist and activist.
Film: The Oscar-winning and star of such films as "This Boy's Life", "What's Eating Gilbert Grape", "The Basketball Diaries"
One: In the 25 years between 1995 and 2020 alone, Leonardo DiCaprio earned north of $300 million from salaries and backend p
Movie: Leonardo's next movie What's Eating Gilbert Grape?
Leonardo: Leonardo DiCaprio is an American actor, producer, philanthropist and activist.
Million: Leonardo DiCaprio has a net worth of $300 million.
Time: Both he and Brad Pitt took paychecks to $10 million a piece (down from $20 million) to appear alongside each other in
Actor: Leonardo DiCaprio is an American actor, producer, philanthropist and activist.
Part: The Oscar-winning and star of such films as "This Boy's Life", "What's Eating Gilbert Grape", "The Basketball Diaries"
Issue: Inspired by the efforts of Al Gore and his campaign against global warming, DiCaprio has opted to amplify his own eff
Diamond: He then earned $20 million a piece for Catch Me If You Can, The Aviator, The Departed, and Blood Diamond.
Issues: Inspired by the efforts of Al Gore and his campaign against global warming, DiCaprio has opted to amplify his own ef
Scorsese: He has starred in several films directed by the legendary Martin Scorsese including Gangs of New York (grossed $19
Hollywood: (Photo by VALERIE MACON/AFP via Getty Images) Leonardo got his start in Hollywood by appearing in a smattering of
Years: In the 25 years between 1995 and 2020 alone, Leonardo DiCaprio earned north of $300 million from salaries and backend
Review: A portrait of an unhappy marriage that falls to pieces, "Revolutionary Road" won favorable reviews from critics, alt
Diamonds: The movie has come under fire from the diamond industry, which insists the
issue of conflict diamonds took place in the 1990s and has been almost
completely eradicated.
Age: Leonardo DiCaprio is 49 years old.
Height: Leonardo DiCaprio is approximately 6 feet tall.
Producer: In 2013, Leonardo DiCaprio collaborated with  Martin Scorsese to star in and co-produce The Wolf of Wall Street.
Oscar: Leonardo DiCaprio received the Oscar for Best Actor for the 2015 film, The Revenant.
Awards: Leonardo DiCaprio's iconic film, Titanic, achieved immense success both critically and commercially. It received an
Titanic: Leonardo DiCaprio's Titanic was the first film to reach the billion dollar mark in international sales.
Star: Leonardo DiCaprio has starred in  Quentin Tarantino's works such as Django Unchained and Once Upon a Time in Hollywood
Perform: In preparation for his role in the 1993 film What’s Eating Gilbert Grape?, Leonardo DiCaprio spent several days stu
Born: Leonardo DiCaprio was born on November 11, 1974, in Los Angeles, California, USA.
Movies: Leonardo DiCaprio has appeared in several well-known films, such as Titanic, Inception, The Revenant, The Wolf of Wa
```