In [36]:

```
#Download the Reuters Corpus
import nltk
nltk.download("reuters")
```

```
[nltk_data] Downloading package reuters to
[nltk_data]     C:\Users\layal\AppData\Roaming\nltk_data...
[nltk_data]   Package reuters is already up-to-date!
```

Out[36]:

```
True
```

In [37]:

```
!pip install matplotlib
!pip install elasticsearch elasticsearch-dsl nltk tqdm

from elasticsearch import Elasticsearch
from elasticsearch.helpers import bulk
from elasticsearch_dsl import Index, Document, Text, Keyword, connections
import nltk
from nltk.corpus import reuters
from sklearn.feature_extraction.text import TfidfVectorizer
import pandas as pd
import matplotlib.pyplot as plt
from wordcloud import WordCloud
import seaborn as sns
from matplotlib.animation import FuncAnimation
```

```
Requirement already satisfied: matplotlib in c:\users\layal\anaconda3
\lib\site-packages (3.5.2)
Requirement already satisfied: fonttools>=4.22.0 in c:\users\layal\ana
conda3\lib\site-packages (from matplotlib) (4.25.0)
Requirement already satisfied: packaging>=20.0 in c:\users\layal\anaco
nda3\lib\site-packages (from matplotlib) (21.3)
Requirement already satisfied: kiwisolver>=1.0.1 in c:\users\layal\ana
conda3\lib\site-packages (from matplotlib) (1.4.2)
Requirement already satisfied: python-dateutil>=2.7 in c:\users\layal
\anaconda3\lib\site-packages (from matplotlib) (2.8.2)
Requirement already satisfied: pyparsing>=2.2.1 in c:\users\layal\anac
onda3\lib\site-packages (from matplotlib) (3.0.9)
Requirement already satisfied: pillow>=6.2.0 in c:\users\layal\anacond
a3\lib\site-packages (from matplotlib) (9.2.0)
Requirement already satisfied: cycler>=0.10 in c:\users\layal\anaconda
3\lib\site-packages (from matplotlib) (0.11.0)
Requirement already satisfied: numpy>=1.17 in c:\users\layal\anaconda3
\lib\site-packages (from matplotlib) (1.21.5)
Requirement already satisfied: six>=1.5 in c:\users\layal\anaconda3\li
b\site-packages (from python-dateutil>=2.7->matplotlib) (1.16.0)
Requirement already satisfied: elasticsearch in c:\users\layal\anacond
a3\lib\site-packages (8.11.0)
Requirement already satisfied: elasticsearch-dsl in c:\users\layal\ana
conda3\lib\site-packages (8.11.0)
Requirement already satisfied: nltk in c:\users\layal\anaconda3\lib\si
te-packages (3.7)
Requirement already satisfied: tqdm in c:\users\layal\anaconda3\lib\si
te-packages (4.64.1)
Requirement already satisfied: elastic-transport<9,>=8 in c:\users\lay
al\anaconda3\lib\site-packages (from elasticsearch) (8.10.0)
Requirement already satisfied: python-dateutil in c:\users\layal\anaco
nda3\lib\site-packages (from elasticsearch-dsl) (2.8.2)
Requirement already satisfied: joblib in c:\users\layal\anaconda3\lib
\site-packages (from nltk) (1.1.0)
Requirement already satisfied: click in c:\users\layal\anaconda3\lib\s
ite-packages (from nltk) (8.0.4)
Requirement already satisfied: regex>=2021.8.3 in c:\users\layal\anaco
nda3\lib\site-packages (from nltk) (2022.7.9)
Requirement already satisfied: colorama in c:\users\layal\anaconda3\li
b\site-packages (from tqdm) (0.4.5)
Requirement already satisfied: urllib3<3,>=1.26.2 in c:\users\layal\an
aconda3\lib\site-packages (from elastic-transport<9,>=8->elasticsearc
h) (1.26.11)
Requirement already satisfied: certifi in c:\users\layal\anaconda3\lib
\site-packages (from elastic-transport<9,>=8->elasticsearch) (2022.9.1
4)
Requirement already satisfied: six>=1.5 in c:\users\layal\anaconda3\li
b\site-packages (from python-dateutil->elasticsearch-dsl) (1.16.0)
```

In [38]:

```
# Download NLTK resources
nltk.download('reuters')
nltk.download('stopwords')
```

```
[nltk_data] Downloading package reuters to
[nltk_data]     C:\Users\layal\AppData\Roaming\nltk_data...
[nltk_data]   Package reuters is already up-to-date!
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\layal\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

Out[38]:

```
True
```

In [39]:

```
# Load Reuters documents
documents = [reuters.raw(file id) for file id in reuters.fileids()]
```

In [40]:

```
# Preprocess the data, Tokenize the text data into words and clean it by
# and performing stemming
from nltk.corpus import stopwords
from nltk.stem import PorterStemmer
from nltk.tokenize import word_tokenize

def preprocess_document(doc):
    stop_words = set(stopwords.words('english'))
    stemmer = PorterStemmer()
    tokens = word_tokenize(doc.lower())
    tokens = [stemmer.stem(word) for word in tokens if word.isalpha() and
    return ' '.join(tokens)

preprocessed documents = [preprocess document(doc) for doc in documents]
```

In [11]:

```
# TF-IDF Vectorization
tfidf_vectorizer = TfidfVectorizer()
tfidf matrix = tfidf vectorizer.fit transform(preprocessed documents)
```

In [41]:

```
# Simple Information Retrieval Function,Implement a function to query the
def retrieve_documents(query, tfidf_matrix, documents):
    query_vec = tfidf_vectorizer.transform([preprocess_document(query)])
    similarities = (query_vec * tfidf_matrix.T).A[0]
    sorted_indices = similarities.argsort()[::-1]
    return [(documents[i][:50], similarities[i]) for i in sorted indices]
```

In [42]:

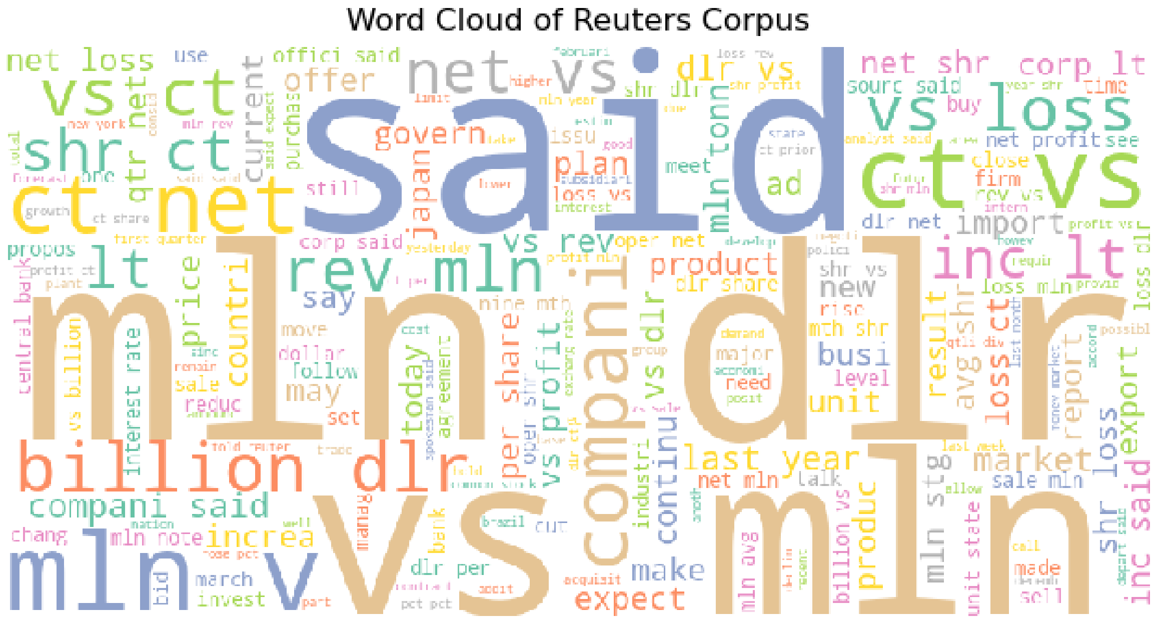
```
# Example of Query
query = "oils"
retrieved_results = retrieve_documents(query, tfidf_matrix, documents)
retrieved_results
```

Out[42]:

```
[('U.S. WARNS OF DEPENDENCE ON FOREIGN OIL\n  A White ', 0.50128565
193787),
 ('U.S. WARNS OF DEPENDENCE ON FOREIGN OIL\n  A White ', 0.48843847
537085666),
 ('ENERGY/FOREIGN INVESTORS\n  Lured by the weakening ', 0.48491067
998259757),
 ('DIVISION SEEN ON HOW TO HELP U.S. OIL INDUSTRY\n  T', 0.44749622
59762729),
 ('WALL STREET STOCKS/U.S. OIL COMPANIES\n  British Pe', 0.43902391
577506206),
 ('SHEARSON LEHMAN UPGRADES U.S. OIL STOCKS\n  Analyst', 0.43365890
800536233),
 ('OILS/FATS STOCKS SEEN FALLING SHARPLY IN 1986/87\n ', 0.43273046
76848475),
 ('IMPERIAL OIL RAISES CRUDE OIL POSTINGS 32 CANADIAN', 0.428058501
11863836),
 ('JAPAN SEES MARGINAL RISE IN EDIBLE OIL DEMAND\n  Th', 0.42315529
412493724),
 ('ENERGY ANALYST PROPOSES U.S. OIL TARIFF\n  Energy a', 0.41651050
```

Visualization 1: Word Cloud of reuters corpus

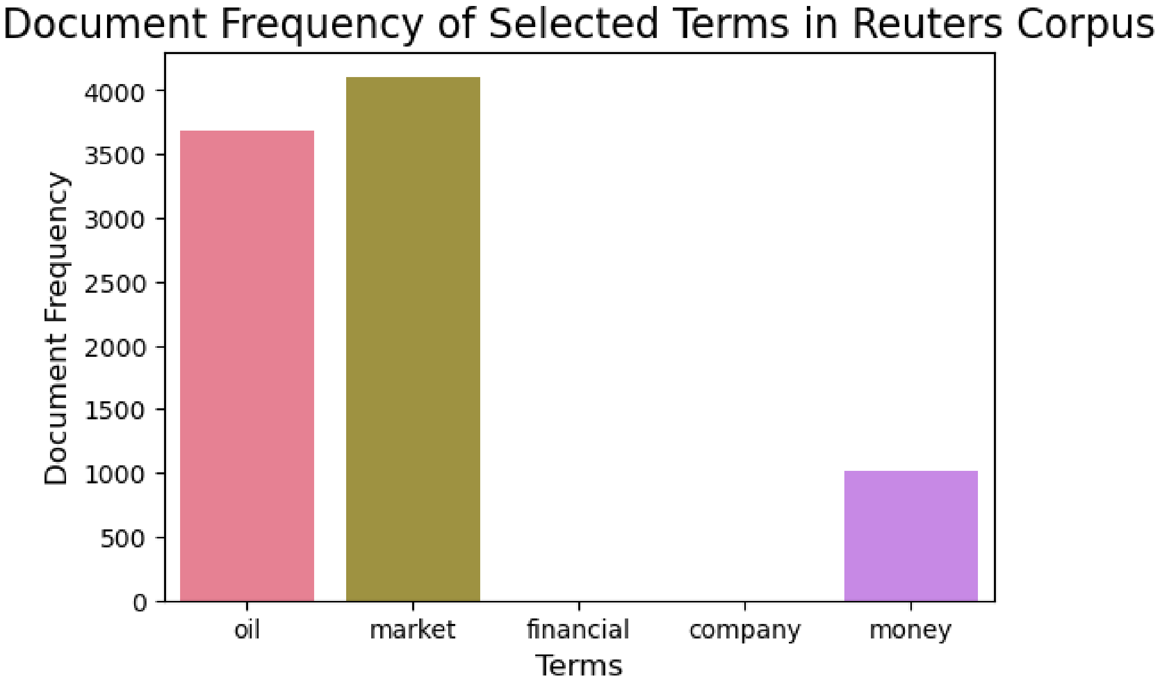
In [16]:
```
all_documents = ' '.join(preprocessed_documents)
wordcloud = WordCloud(width=800, height=400, background_color='white', cc
plt.figure(figsize=(8, 5))
plt.imshow(wordcloud, interpolation='nearest')
plt.axis('off')
plt.title('Word Cloud of Reuters Corpus')
plt.show()
```



Visualization 2: Bar Chart - Document Frequency of Selected Terms

In [19]:

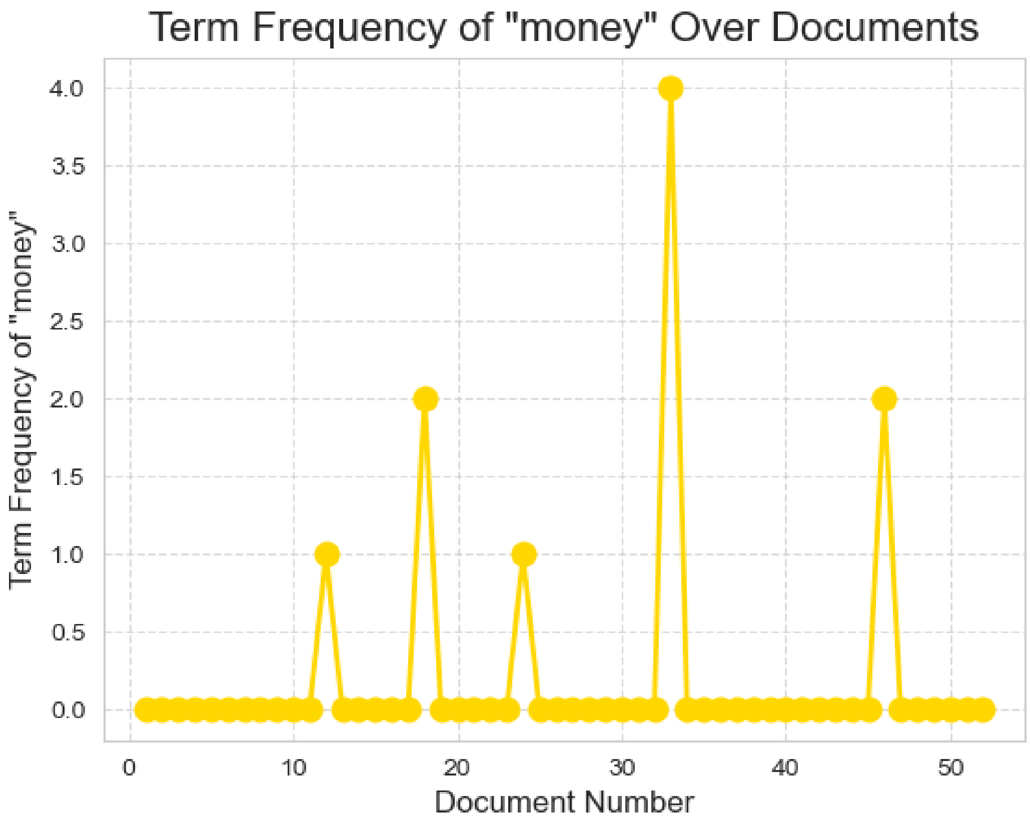
```
selected_terms = ['oil', 'market', 'financial', 'company', 'money']
term_frequencies = [sum(doc.count(term) for doc in preprocessed_documents
                   ]
custom_palette = sns.color_palette("husl", len(selected_terms))
plt.figure(figsize=(6, 4))
sns.barplot(x=selected_terms, y=term_frequencies, palette=custom_palette)
plt.xlabel('Terms', fontsize=12)
plt.ylabel('Document Frequency', fontsize=12)
plt.title('Document Frequency of Selected Terms in Reuters Corpus', fonts
plt.show()
```



Visualization 3: animated line chart , visualize the frequency of the term in each document.

In [48]:

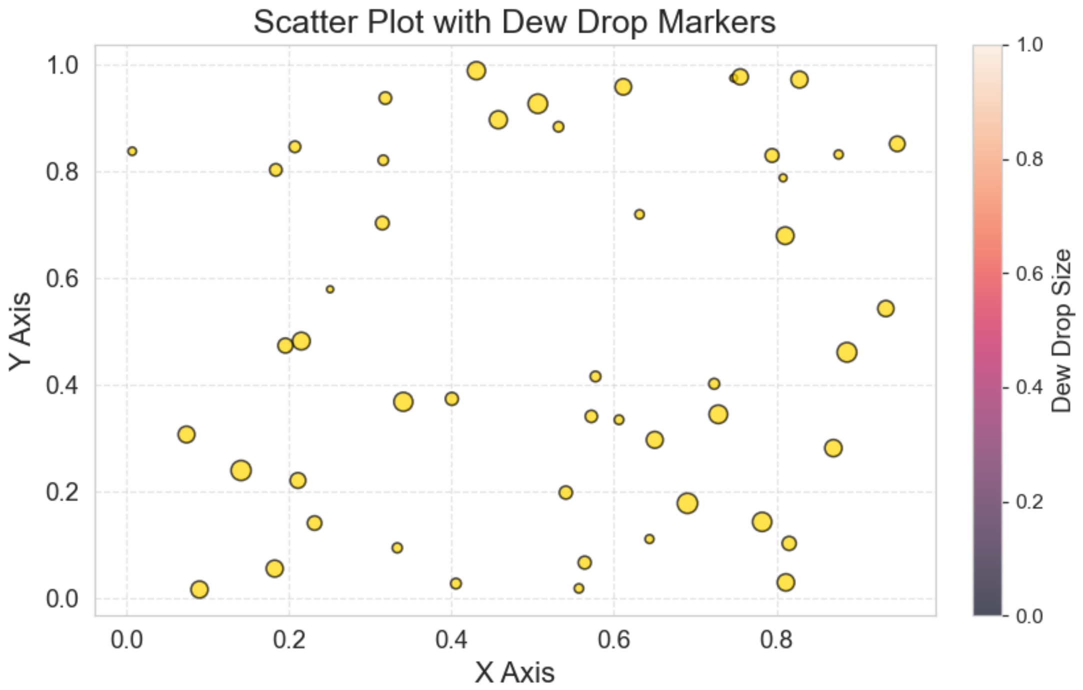
```
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.animation import FuncAnimation
import seaborn as sns
sns.set_style("whitegrid")
# Query term
query_term = 'money'

def update(frame):
    plt.clf()
    term_frequency = [doc.count(query_term) for doc in preprocessed_docum
    plt.plot(range(1, frame+2), term_frequency, marker='o', color='gold',
    plt.xlabel('Document Number', fontsize=12)
    plt.ylabel(f'Term Frequency of "{query_term}"', fontsize=12)
    plt.title(f'Term Frequency of "{query_term}" Over Documents', fontsiz
    plt.xticks(fontsize=10)
    plt.yticks(fontsize=10)
    plt.grid(axis='both', linestyle='--', alpha=0.7)
    plt.pause(0.1)

# Set up the plot
plt.figure(figsize=(8, 5))
plt.tight_layout()

# Enable interactive mode
%matplotlib notebook

ani = FuncAnimation(plt.gcf(), update, frames=len(preprocessed_documents)

# Show the animated line chart
plt.show()
```



# Visualization 4: scatter plot of two term

In [47]:

```
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

# Sample data
x_values = np.random.rand(50)
y_values = np.random.rand(50)
dew_point_sizes = np.random.randint(10, 100, size=50)  # Sizes for each c

sns.set_palette("pastel")

# Create a scatter plot with custom markers (dew drops)
plt.figure(figsize=(8, 5))
scatter = plt.scatter(x_values, y_values, s=dew_point_sizes, c='gold', al

# Set axis labels and title
plt.xlabel('X Axis', fontsize=14)
plt.ylabel('Y Axis', fontsize=14)
plt.title('Scatter Plot with Dew Drop Markers', fontsize=16)

# Customize colorbar for size legend
size_legend = plt.colorbar(scatter, orientation='vertical', fraction=0.04
size_legend.set_label('Dew Drop Size', fontsize=12)

# Customize tick labels
plt.xticks(fontsize=12)
plt.yticks(fontsize=12)

# Add grid lines
plt.grid(True, linestyle='--', alpha=0.5)

plt.show()
```



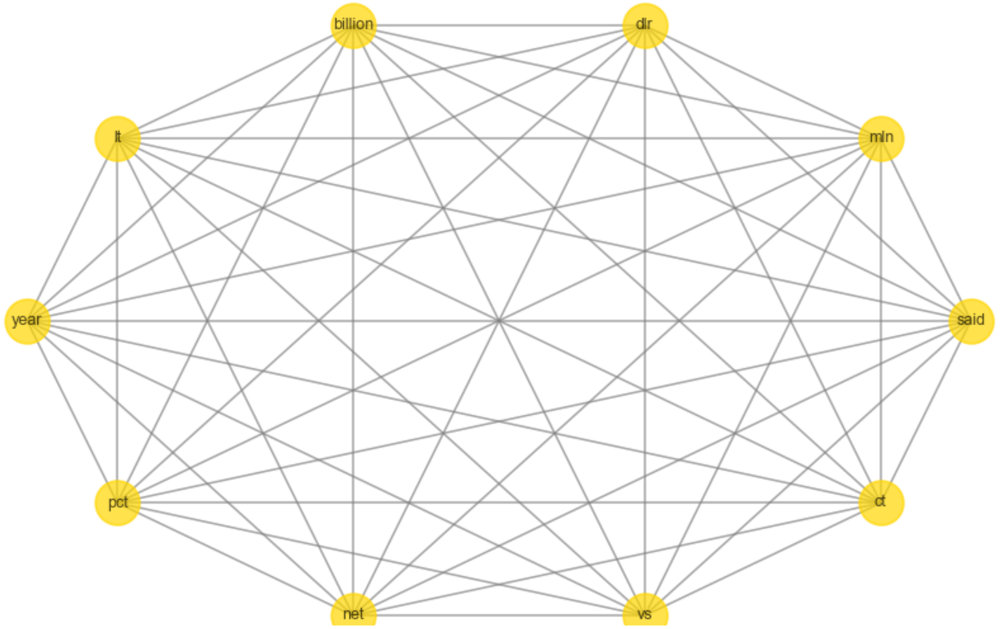
Visualization 5: network graph

```
In [44]: !pip install networkx
```

```
Requirement already satisfied: networkx in c:\users\layal\anaconda3\li
b\site-packages (2.8.4)
```

In [45]:

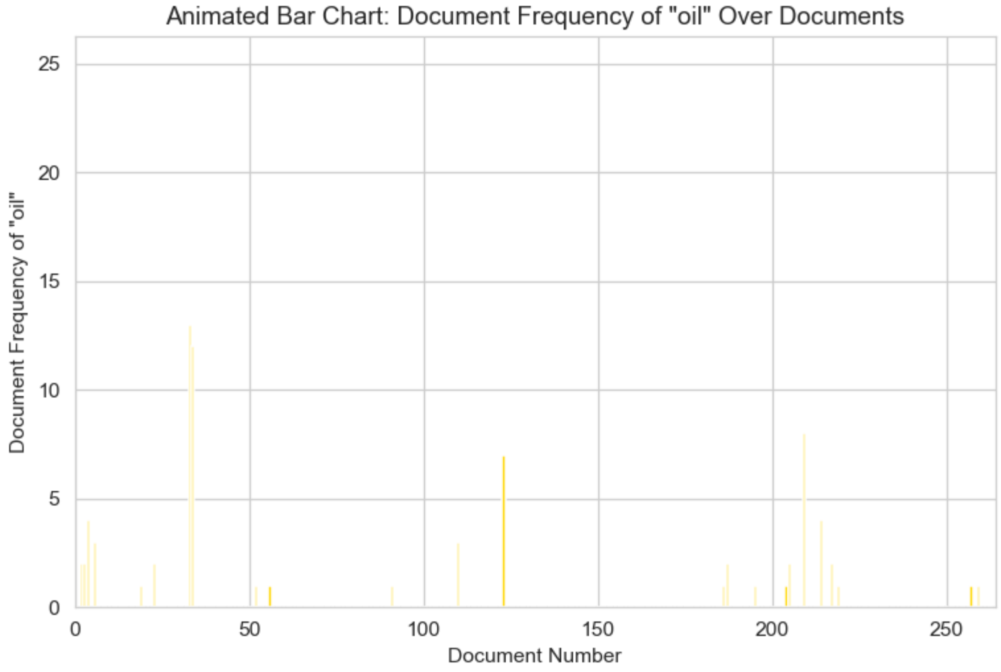
```
import networkx as nx
from sklearn.feature_extraction.text import CountVectorizer

# Choose the number of top terms to include in the graph
top_terms_count = 10

# Extract the top terms based on document frequency
vectorizer = CountVectorizer()
term_matrix = vectorizer.fit_transform(preprocessed_documents)
term_frequencies = term_matrix.sum(axis=0)
top_term_indices = np.argsort(term_frequencies)[0, -top_terms_count:][::-
top_terms = [term for term, idx in vectorizer.vocabulary_.items() if idx

# Create a co-occurrence matrix
co_occurrence_matrix = term_matrix.T @ term_matrix
co_occurrence_matrix.setdiag(0)

# Create a network graph
G = nx.Graph()

# Add nodes to the graph
for term in top_terms:
    G.add_node(term)

# Add edges to the graph based on co-occurrence
edges = [(term1, term2, {'weight': co_occurrence_matrix[vectorizer.vocabu
         for term1 in top_terms for term2 in top_terms if term1 != term2
         and co_occurrence_matrix[vectorizer.vocabulary_[term1], vectoriz
G.add_edges_from(edges)

# Visualize the network graph
plt.figure(figsize=(8, 5))
pos = nx.circular_layout(G)
nx.draw(G, pos, with_labels=True, font_size=8, font_color='black', node_s
        node_color='gold', edge_color='grey', width=1, alpha=0.7)
plt.title('Co-occurrence Network Graph of Top Terms')
plt.show()
```



Visualization 6: animated bar chart

In [43]:

```
import numpy as np
from matplotlib.animation import FuncAnimation
import matplotlib.pyplot as plt

# Enable interactive mode
%matplotlib notebook
plt.ion()

# the term for the animated bar chart
animated_term = 'oil'

# Get the document frequencies of the term across documents
term_document_frequencies = [doc.count(animated_term) for doc in preproce

# Create a bar chart
fig, ax = plt.subplots(figsize=(8, 5))
bars = ax.bar(range(len(preprocessed_documents)), term_document_frequenci
ax.set_xlabel('Document Number')
ax.set_ylabel(f'Document Frequency of "{animated_term}"')
ax.set_title(f'Animated Bar Chart: Document Frequency of "{animated_term}

def update(frame):
    # Update the heights of the bars for each frame
    for bar, height in zip(bars, term_document_frequencies[:frame+1]):
        bar.set_height(height)
    ax.set_xlim(0, frame + 1)  # Adjust the x-axis limit
    plt.pause(0.1)

ani = FuncAnimation(fig, update, frames=len(preprocessed_documents), repe
plt.show()
```



In [ ]: