```python
import pandas as pd
from nltk import ngrams
from sklearn.decomposition import PCA
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score
import matplotlib.pyplot as plt
import re
import numpy as np

# Read the comments data
df = pd.read_csv('Comments_Data.csv')

def generate_ngrams(text, n, target_words):
    sentences = re.split(r'[.!?]', text)  # Split text into sentences
    n_grams = []
    for sentence in sentences:
        words = sentence.split()
        for i in range(len(words)):
            if words[i] in target_words:
                start = max(i - n, 0)
                end = min(i + n + 1, len(words))
                context = words[start:i] + words[i+1:end]
                n_grams.append(' '.join(context))
    return n_grams

# Concatenate all comments into a single string
all_comments = ' '.join(df['comment_body'])

target_words = ['investor', 'FDIC', 'startup', 'government', 'deposit', 'bailout']
colors = ['lightcoral', 'lightskyblue', 'lightgreen', 'red', 'orange', 'darkgreen']
n = 2  # Adjust the context window size as needed

# Generate 2-grams for each target word
target_ngrams = {}
for word in target_words:
    word_ngrams = generate_ngrams(all_comments, n, [word])
    target_ngrams[word] = word_ngrams

# Combine all 2-grams into a single list
all_2grams = [gram for ngrams in target_ngrams.values() for gram in ngrams]

# Apply CountVectorizer to convert 2-grams into a matrix of token counts
vectorizer = CountVectorizer()
X = vectorizer.fit_transform(all_2grams)

# Get the feature names
feature_names = vectorizer.get_feature_names_out()

# Apply PCA to the token count matrix
pca = PCA(n_components=2)
X_pca = pca.fit_transform(X.toarray())

# Perform k-means clustering with k=5
kmeans = KMeans(n_clusters=7, random_state=0).fit(X_pca)

# Plot the PCA results with cluster separation using colors
plt.figure(figsize=(8, 6))
for i, word in enumerate(target_words):
    start_index = sum(len(target_ngrams[w]) for w in target_words[:i])
    end_index = start_index + len(target_ngrams[word])
    plt.scatter(X_pca[start_index:end_index, 0],
                X_pca[start_index:end_index, 1],
                c=colors[i],
                label=word)
plt.xlabel('Principal Component 1')
plt.ylabel('Principal Component 2')
plt.title('PCA Visualization of 2-grams with Cluster Separation (K-Means)')
plt.legend()
plt.show()
```