In [1]:
```
import nltk
nltk.download('popular')
import demoji
from wordcloud import WordCloud
from nltk import word_tokenize
from nltk.corpus import stopwords
from nltk.util import bigrams
from nltk import FreqDist
import spacy
import string
```

```
[nltk_data] Downloading collection 'popular'
[nltk_data]    |
[nltk_data]    | Downloading package cmudict to
[nltk_data]    |     /Users/shireesh/nltk_data...
[nltk_data]    |   Package cmudict is already up-to-date!
[nltk_data]    | Downloading package gazetteers to
[nltk_data]    |     /Users/shireesh/nltk_data...
[nltk_data]    |   Package gazetteers is already up-to-date!
[nltk_data]    | Downloading package genesis to
[nltk_data]    |     /Users/shireesh/nltk_data...
[nltk_data]    |   Package genesis is already up-to-date!
[nltk_data]    | Downloading package gutenberg to
[nltk_data]    |     /Users/shireesh/nltk_data...
[nltk_data]    |   Package gutenberg is already up-to-date!
[nltk_data]    | Downloading package inaugural to
[nltk_data]    |     /Users/shireesh/nltk_data...
[nltk_data]    |   Package inaugural is already up-to-date!
[nltk_data]    | Downloading package movie_reviews to
[nltk_data]    |     /Users/shireesh/nltk_data...
[nltk_data]    |   Package movie_reviews is already up-to-date!
[nltk_data]    | Downloading package names to
[nltk_data]    |     /Users/shireesh/nltk_data...
[nltk_data]    |   Package names is already up-to-date!
[nltk_data]    | Downloading package shakespeare to
[nltk_data]    |     /Users/shireesh/nltk_data...
[nltk_data]    |   Package shakespeare is already up-to-date!
[nltk_data]    | Downloading package stopwords to
[nltk_data]    |     /Users/shireesh/nltk_data...
[nltk_data]    |   Package stopwords is already up-to-date!
[nltk_data]    | Downloading package treebank to
[nltk_data]    |     /Users/shireesh/nltk_data...
[nltk_data]    |   Package treebank is already up-to-date!
[nltk_data]    | Downloading package twitter_samples to
[nltk_data]    |     /Users/shireesh/nltk_data...
[nltk_data]    |   Package twitter_samples is already up-to-date!
[nltk_data]    | Downloading package omw to
[nltk_data]    |     /Users/shireesh/nltk_data...
[nltk_data]    |   Package omw is already up-to-date!
[nltk_data]    | Downloading package omw-1.4 to
[nltk_data]    |     /Users/shireesh/nltk_data...
[nltk_data]    |   Package omw-1.4 is already up-to-date!
[nltk_data]    | Downloading package wordnet to
[nltk_data]    |     /Users/shireesh/nltk_data...
[nltk_data]    |   Package wordnet is already up-to-date!
[nltk_data]    | Downloading package wordnet2021 to
[nltk_data]    |     /Users/shireesh/nltk_data...
[nltk_data]    |   Package wordnet2021 is already up-to-date!
[nltk_data]    | Downloading package wordnet31 to
[nltk_data]    |     /Users/shireesh/nltk_data...
[nltk_data]    |   Package wordnet31 is already up-to-date!
[nltk_data]    | Downloading package wordnet_ic to
[nltk_data]    |     /Users/shireesh/nltk_data...
[nltk_data]    |   Package wordnet_ic is already up-to-date!
[nltk_data]    | Downloading package words to
[nltk_data]    |     /Users/shireesh/nltk_data...
[nltk_data]    |   Package words is already up-to-date!
[nltk_data]    | Downloading package maxent_ne_chunker to
[nltk_data]    |     /Users/shireesh/nltk_data...
[nltk_data]    |   Package maxent_ne_chunker is already up-to-date!
[nltk_data]    | Downloading package punkt to
[nltk_data]    |     /Users/shireesh/nltk_data...
[nltk_data]    |   Package punkt is already up-to-date!
[nltk_data]    | Downloading package snowball_data to
[nltk_data]    |     /Users/shireesh/nltk_data...
[nltk_data]    |   Package snowball_data is already up-to-date!
[nltk_data]    | Downloading package averaged_perceptron_tagger to
[nltk_data]    |     /Users/shireesh/nltk_data...
[nltk_data]    |   Package averaged_perceptron_tagger is already up-
[nltk_data]    |       to-date!
[nltk_data]    |
[nltk_data]  Done downloading collection popular
```

In [2]:
```
import os
import glob
def collect_data():
    text_file_pattern = "*.txt"  # You can adjust the pattern to match your file extensions
    text_files = glob.glob(os.path.join("../nhs/content", text_file_pattern))
    data = {}
    for file_path in text_files:
        with open(file_path, 'r', encoding='utf-8') as file:
            file_name = os.path.basename(file_path)
            file_content = file.read()
            data[file_name] = file_content
    return data
```

In [3]:
```
corpus = collect_data()
```

In [4]:
```
text = ""
for data in corpus:
    text += " " + data
```

In [5]:
```
def remove_punctuation(text):
    # Create a translation table to remove punctuation
    translator = str.maketrans('', '', string.punctuation)
    # Use translate method to remove punctuation
    cleaned_text = text.translate(translator)
    return cleaned_text

def remove_stop_words(text):
    nltk.download("stopwords")
    nltk_stopwords = stopwords.words('english')

    nlp = spacy.load("en_core_web_sm")
    spacy_stopwords = nlp.Defaults.stop_words

    stop_words = (*nltk_stopwords, *spacy_stopwords, "NHStxt")

    tokens = word_tokenize(text)
    tokens = [token for token in tokens if token not in stop_words]
    return " ".join(tokens)

def remove_emoji_and_smart_quotes(text):
    # replacing emojis with description
    text = demoji.replace_with_desc(text)
    #Removing smart quotes
    return text.replace("”", "\"").replace("“","\"")

def clean_text(text):
    text = remove_punctuation(text)
    text = remove_emoji_and_smart_quotes(text)
    return remove_stop_words(text)
```

1. Remove punctuation and stop words in the data files. (10 points)

In [6]:
```
text = clean_text(text)
```

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

In [7]:
```
text
```

Out[7]:
```
'Food colours hyperactivity Lung health checks Carotid endarterectomy Middle East respiratory syndrome MERS Kidney
infection Autism NHS screening Bedwetting children Overview kidney cancer Nail problems CreutzfeldtJakob disease Ao
rtic valve replacement Soft tissue sarcomas Clostridium difficile C diff Circumcision men Nonalcoholic fatty liver
disease NAFLD Trichomoniasis TENS transcutaneous electrical nerve stimulation CharcotMarieTooth disease Body dysmor
phic disorder BDD Heart failure Heartburn acid reflux Pityriasis rosea Astigmatism Attention deficit hyperactivity
disorder ADHD Frozen shoulder Cardiovascular disease Chilblains Low sperm count Tonguetie Eye tests children Sjögre
ns syndrome Thumb pain Smelly urine Teeth grinding bruxism Myositis polymyositis dermatomyositis NHS Molluscum cont
agiosum Anxiety disorders children Bartholins cyst Hair loss Angioedema IVF Botulism Back pain Rhesus disease Pelvi
c inflammatory disease Epidermolysis bullosa Erythema nodosum Salivary gland stones Chlamydia Tics Cyclospora Spina
l muscular atrophy Night sweats Phobias Stop smoking treatments Muscular dystrophy intrauterine insemination IUI NH
S Fabricated induced illness Hives Postmortem Dialysis Anticoagulant medicines Bulging eyes exophthalmos Klinefelte
r syndrome Pneumonia Respiratory tract infections RTIs Cirrhosis Itchy skin Hernia Breathholding babies children Pl
antar fasciitis Ileostomy Smelly feet Heart attack Cardiac catheterisation coronary angiography Soiling child pooin
g pants PMS premenstrual syndrome Hallucinations hearing voices Polio Broken finger thumb Ovarian cyst PraderWilli
syndrome Jellyfish sea creature stings Temporomandibular disorder Giardiasis Gastrectomy Brugada syndrome Lyme dise
ase Liver cancer Anal cancer Genital herpes Spleen problems spleen removal Stye Differences sex development Hoardin
g disorder Problems swallowing pills Achalasia Otosclerosis Chronic obstructive pulmonary disease COPD Local anaest
hesia Period pain Intracranial hypertension Febrile seizures Pain ball foot Necrotising fasciitis Toxocariasis Tinn
itus Huntingtons disease Adenomyosis Cuts grazes What ileostomy Vaginal cancer Urinary incontinence Femoral hernia
```

2. Print out the 20 most common words in the frequency distribution. (10 points)

In [8]:
```
def get_most_frequent_words(text, count=20):
    # Tokenize the text
    tokens = word_tokenize(text)
    # Calculate word frequencies
    freq_dist = FreqDist(tokens)
    # Get the 10 most frequent words
    return freq_dist.most_common(count)
```

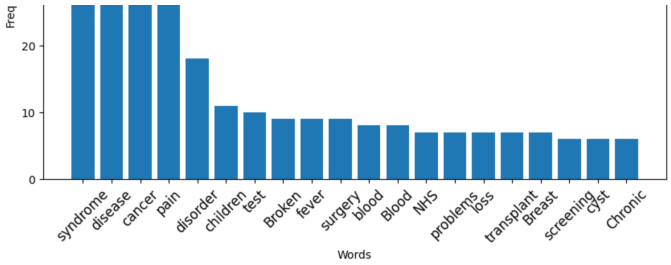
In [9]:
```
get_most_frequent_words(text)
```

Out[9]:
```
[('syndrome', 53),
 ('disease', 42),
 ('cancer', 34),
 ('pain', 29),
 ('disorder', 18),
 ('children', 11),
 ('test', 10),
 ('Broken', 9),
 ('fever', 9),
 ('surgery', 9),
 ('blood', 8),
 ('Blood', 8),
 ('NHS', 7),
 ('problems', 7),
 ('loss', 7),
 ('transplant', 7),
 ('Breast', 7),
 ('screening', 6),
 ('cyst', 6),
 ('Chronic', 6)]
```

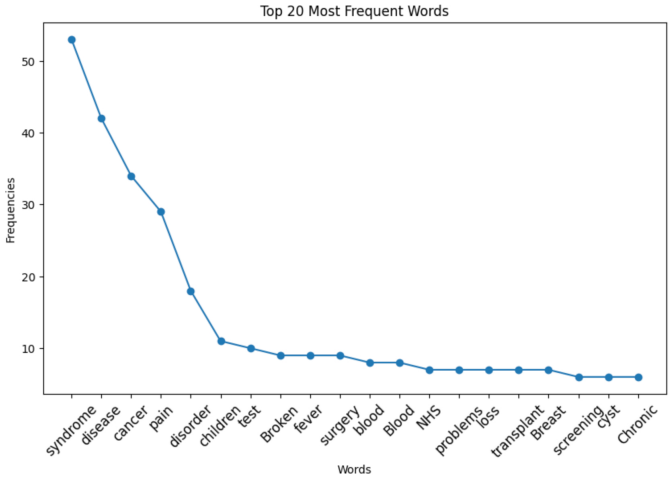
3. Plot the frequency distribution in three different ways and explain which one is most effective. (40 points)

In [10]:
```
import matplotlib.pyplot as plt
def plot_bar_chart_frequency(most_common_words, prefix="3-"):
    # Extract words and their frequencies
    words, frequencies = zip(*most_common_words)

    # Create a bar chart
    plt.figure(figsize=(10, 6))
    plt.bar(words, frequencies)
    plt.xlabel('Words')
    plt.ylabel('Frequencies')
    plt.title('Top 20 Most Frequent Words')
    plt.xticks(rotation=45, fontsize=12)

    # Save the chart as an image file
    plt.savefig(prefix+'week_6_bar_frequencies.png', bbox_inches='tight')

    # Show the chart (optional)
    plt.show()


def plot_line_chart_frequency(most_common_words, prefix="3-"):
    # Extract words and their frequencies
    words, frequencies = zip(*most_common_words)

    # Create a bar chart
    plt.figure(figsize=(10, 6))
    plt.plot(words, frequencies, marker='o', linestyle='-')
    plt.xlabel('Words')
    plt.ylabel('Frequencies')
    plt.title('Top 20 Most Frequent Words')
    plt.xticks(rotation=45, fontsize=12)

    # Save the chart as an image file
    plt.savefig(prefix+'week_6_line_frequencies.png', bbox_inches='tight')

    # Show the chart (optional)
    plt.show()

def plot_word_cloud_frequency(most_common_words,prefix="3-"):
    # Create a dictionary from the most common words
    wordcloud_dict = dict(most_common_words)
    # Generate a word cloud from the frequency distribution
    wordcloud = WordCloud(width=800, height=400, background_color='white').generate_from_frequencies(wordcloud_dict)

    # Create a figure for the word cloud
    plt.figure(figsize=(10, 6))
    plt.imshow(wordcloud, interpolation='bilinear')
    plt.axis('off')
    plt.title('Word Cloud')

    # Save the word cloud as an image file
    wordcloud.to_file(prefix+'week5_word_cloud.png')

    # Show the word cloud (optional)
    plt.show()
```

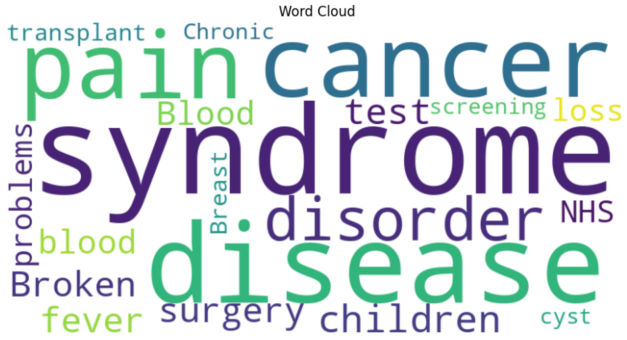
In [11]:
```
most_common_words = get_most_frequent_words(text)
plot_bar_chart_frequency(most_common_words)
```



In [12]:
```
plot_line_chart_frequency(most_common_words)
```



In [13]:
```
plot_word_cloud_frequency(most_common_words)
```



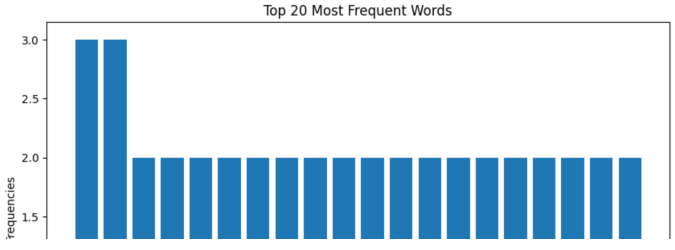
4. Count and display the most common 20 bigrams in your data. (10 points)

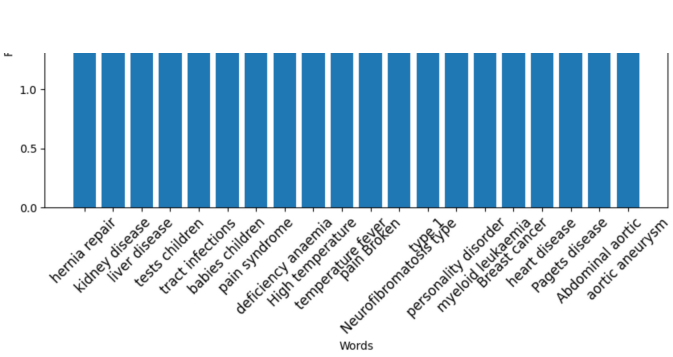
In [14]:
```
def get_most_common_bigrams(text):
    # Tokenize the text into words
    words = word_tokenize(text)
    # Generate bigrams
    bi_grams = list(bigrams(words))
    # Calculate the frequency distribution of bigrams
    bi_gram_freq = FreqDist(bi_grams)
    # Get the 20 most common bigrams
    most_common_bigrams = bi_gram_freq.most_common(20)
    return most_common_bigrams
```

In [15]:
```
get_most_common_bigrams(text)
```

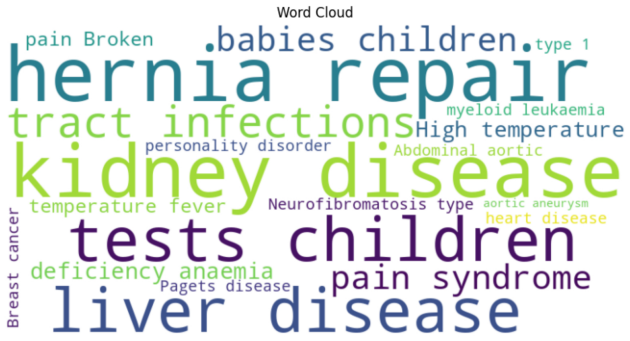
Out[15]:
```
[(('hernia', 'repair'), 3),
 (('kidney', 'disease'), 3),
 (('liver', 'disease'), 2),
 (('tests', 'children'), 2),
 (('tract', 'infections'), 2),
 (('babies', 'children'), 2),
 (('pain', 'syndrome'), 2),
 (('deficiency', 'anaemia'), 2),
 (('High', 'temperature'), 2),
 (('temperature', 'fever'), 2),
 (('pain', 'Broken'), 2),
 (('Neurofibromatosis', 'type'), 2),
 (('type', '1'), 2),
 (('personality', 'disorder'), 2),
 (('myeloid', 'leukaemia'), 2),
 (('Breast', 'cancer'), 2),
 (('heart', 'disease'), 2),
 (('Pagets', 'disease'), 2),
 (('Abdominal', 'aortic'), 2),
 (('aortic', 'aneurysm'), 2)]
```

In [16]:
```
bigrams = [(i+' '+j, q) for (i,j), q in get_most_common_bigrams(text)]
plot_bar_chart_frequency(bigrams, prefix="4-")
```





In [17]:
```
plot_word_cloud_frequency(bigrams, prefix="4-")
```



5. Propose and implement an effective way to measure the similarities among documents/records that exist in your data. (30 points)

In [18]:
```
all_text = clean_text(text)
words = word_tokenize(all_text)
word_features = list(FreqDist(words).most_common(1000))
word_features = [i for i,j in word_features]
```

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

In [19]:
```
def get_features(text):
    features = {}
    for word in word_features:
        features[word] = 1 if word in text else 0
    return features
```

In [20]:
```
dataframe_data = []
for key in corpus.keys():
    features = get_features(corpus[key])
    features['title'] = key
    dataframe_data.append(features)

dataframe_data = {idx: dataframe_data[idx] for idx in range(len(dataframe_data))}
```

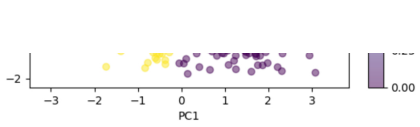
In [21]:
```
import pandas as pd
df = pd.DataFrame.from_dict(data=dataframe_data, orient='index', columns=word_features)
```

In [22]:
```
df.columns
```

Out[22]:
```
Index(['syndrome', 'disease', 'cancer', 'pain', 'disorder', 'children', 'test',
       'Broken', 'fever', 'surgery',
       ...
       'MECFS', 'Malignant', 'Anabolic', 'steroid', 'Pataus', 'Vaccinations',
       'Lipoedema', 'Q', 'cholesterol', 'Antifungal'],
      dtype='object', length=1000)
```

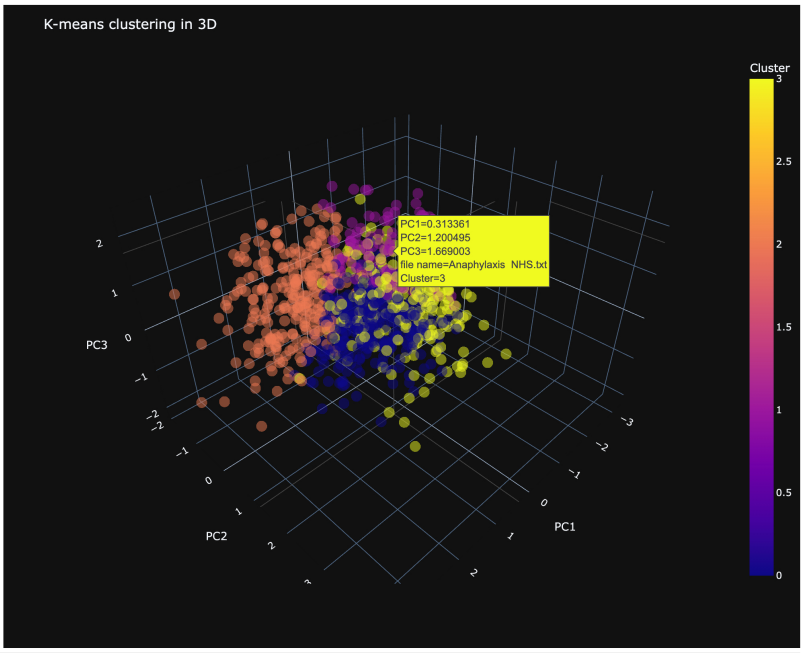
In [23]:
```
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.decomposition import PCA
from sklearn.cluster import KMeans

# Perform k-means clustering with k=3
kmeans = KMeans(n_clusters=3, random_state=42).fit(df)

# Get the cluster labels
cluster_labels = kmeans.labels_

# Use PCA to reduce to 2 dimensions
pca = PCA(n_components=2)
pca_features = pca.fit_transform(df.values)
pca_df = pd.DataFrame(
    data=pca_features,
    columns=['PC1', 'PC2'])

# plot data
plt.scatter(pca_df["PC1"], pca_df["PC2"], c=cluster_labels, cmap='viridis', alpha=0.5)
plt.xlabel("PC1")
plt.ylabel("PC2")
plt.title("K-means clustering")
plt.colorbar()
plt.savefig("clustering_2d.png")
plt.show()
```

```
/Users/shireesh/opt/anaconda3/envs/COMP293/lib/python3.10/site-packages/sklearn/cluster/_kmeans.py:1416: FutureWarnin
g: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppre
ss the warning
  super()._check_params_vs_input(X, default_n_init=10)
```



In [24]:
```
import plotly.express as px
# Perform k-means clustering with k=3
kmeans = KMeans(n_clusters=4, random_state=42).fit(df)

# Get the cluster labels
cluster_labels = kmeans.labels_

# Use PCA to reduce to 3 dimensions
pca = PCA(n_components=3)
pca_features = pca.fit_transform(df.values)
pca_df = pd.DataFrame(
    data=pca_features,
    columns=['PC1', 'PC2', 'PC3'])

# Add cluster labels, gene names, and cell types to PCA dataframe
pca_df['Cluster'] = cluster_labels
pca_df['file name'] = corpus.keys()

# Create 3D scatter plot
fig = px.scatter_3d(pca_df, x='PC1', y='PC2', z='PC3', color='Cluster', opacity=0.5,width=1000, height=800, hover_data=

# Set title and axis labels
fig.update_layout(title='K-means clustering in 3D', scene=dict(xaxis_title='PC1', yaxis_title='PC2', zaxis_title='PC3')

# Show the plot
fig.show()
```

```
/Users/shireesh/opt/anaconda3/envs/COMP293/lib/python3.10/site-packages/sklearn/cluster/_kmeans.py:1416: FutureWarnin
g: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppre
ss the warning
  super()._check_params_vs_input(X, default_n_init=10)
```



In [ ]: