Abhishek Murthy
21BDS0064
Fall Sem 2024-2025
DA-5
Data Mining Lab
22-10-2024

In [1]:
```
from sklearn.feature_extraction.text import CountVectorizer
from scipy.spatial.distance import jaccard as scipy_jaccard
import math

documents = [
"ant ant bee", # d1
"dog bee dog hog dog ant dog", # d2
"cat gnu dog eel fox" # d3
]

vectorizer = CountVectorizer()
X = vectorizer.fit_transform(documents).toarray()
feature_names = vectorizer.get_feature_names_out()
for i, doc_vector in enumerate(X):
  print(f"Document {i + 1} vector: {doc_vector}")
  print("Corresponding words:")
  for j, count in enumerate(doc_vector):
    if count > 0:
      print(f"{feature_names[j]}: {count}")
  print()
```

```
Document 1 vector: [2 1 0 0 0 0 0 0]
Corresponding words:
ant: 2
bee: 1

Document 2 vector: [1 1 0 4 0 0 0 1]
Corresponding words:
ant: 1
bee: 1
dog: 4
hog: 1

Document 3 vector: [0 0 1 1 1 1 1 0]
Corresponding words:
cat: 1
dog: 1
eel: 1
fox: 1
gnu: 1
```

In [2]:
```
cosine_sim = []
for i in range(len(X)):
  row = []
  for j in range(len(X)):
    if i == j:
      row.append(1) # Similarity of a document with itself is 1
    else:
      dot_product = sum(X[i] * X[j])
      norm_i = sum(X[i] ** 2) ** 0.5
      norm_j = sum(X[j] ** 2) ** 0.5
      cosine_similarity = dot_product / (norm_i * norm_j) if (norm_i *
      norm_j) != 0 else 0
      row.append(cosine_similarity)
  cosine_sim.append(row)
print("Cosine Similarity:\n", cosine_sim)
```

```
Cosine Similarity:
 [[1, 0.3077935056255462, 0.0], [0.3077935056255462, 1, 0.4103913408340616], [0.0,
0.4103913408340616, 1]]
```

In [3]:
```
binary_X = (X > 0).astype(int)
jaccard_dist = []
for i in range(len(binary_X)):
  row = []
  for j in range(len(binary_X)):
    if i == j:
      row.append(0)
    else:
      intersection = sum((binary_X[i] & binary_X[j]))
      union = sum((binary_X[i] | binary_X[j]))
      row.append(1 - (intersection / union))
  jaccard_dist.append(row)
print("Jaccard Distance:\n", jaccard_dist)
```

```
Jaccard Distance:
 [[0, 0.5, 1.0], [0.5, 0, 0.875], [1.0, 0.875, 0]]
```

In [4]:
```
euclidean_dist = []
for i in range(len(X)):
  row = []
  for j in range(len(X)):
    if i == j:
      row.append(0) # Distance to itself is 0
    else:
      distance = math.sqrt(sum((X[i] - X[j]) ** 2))
      row.append(distance)
  euclidean_dist.append(row)
print("Euclidean Distance:\n", euclidean_dist)
```

```
Euclidean Distance:
 [[0, 4.242640687119285, 3.1622776601683795], [4.242640687119285, 0, 4.0], [3.1622
776601683795, 4.0, 0]]
```

In [5]:
```
most_similar_docs_cosine = (0, 1)
max_similarity = -1
for i in range(len(cosine_sim)):
  for j in range(len(cosine_sim)):
    if i != j and cosine_sim[i][j] > max_similarity:
      max_similarity = cosine_sim[i][j]
      most_similar_docs_cosine = (i, j)
print(f"Most similar documents based on Cosine Similarity:d{most_similar_docs_cosir
```

```
Most similar documents based on Cosine Similarity:d2 and d3
```

In [6]:
```
most_similar_docs_jaccard = (0, 1)
min_jaccard_distance = float('inf')
for i in range(len(jaccard_dist)):
  for j in range(len(jaccard_dist)):
    if i != j and jaccard_dist[i][j] < min_jaccard_distance:
      min_jaccard_distance = jaccard_dist[i][j]
      most_similar_docs_jaccard = (i, j)
print(f"Most similar documents based on Jaccard Similarity: d{most_similar_docs_jac
```

```
Most similar documents based on Jaccard Similarity: d1 and d2
```

In [7]:
```
most_similar_docs_euclidean = (0, 1)
min_euclidean_distance = float('inf')
for i in range(len(euclidean_dist)):
  for j in range(len(euclidean_dist)):
    if i != j and euclidean_dist[i][j] < min_euclidean_distance:
      min_euclidean_distance = euclidean_dist[i][j]
      most_similar_docs_euclidean = (i, j)
print(f"Most similar documents based on Euclidean Distance: d{most_similar_docs_euc
```

```
Most similar documents based on Euclidean Distance: d1 and d3
```