Banking Chat Intent Clustering Project

```
!huggingface-cli login
```

```
⚠️ Warning: 'huggingface-cli login' is deprecated. Use 'hf auth login' instead.

    _|    _|  _|    _|    _|_|_|    _|_|_|  _|_|_|  _|      _|    _|_|_|      _|_|_|_|    _|_|      _|_|_|  _|_|_|_|
    _|    _|  _|    _|  _|        _|          _|    _|_|    _|  _|            _|        _|    _|  _|        _|
    _|_|_|_|  _|    _|  _|  _|_|  _|  _|_|    _|    _|  _|  _|  _|  _|_|      _|_|_|    _|_|_|_|  _|        _|_|_|
    _|    _|  _|    _|  _|    _|  _|    _|    _|    _|    _|_|  _|    _|      _|        _|    _|  _|        _|
    _|    _|    _|_|      _|_|_|    _|_|_|  _|_|_|  _|      _|    _|_|_|      _|        _|    _|    _|_|_|  _|_|_|_|

    To log in, `huggingface_hub` requires a token generated from https://huggingface.co/settings/tokens .
Enter your token (input will not be visible):
Add token as git credential? (Y/n) Y
Token is valid (permission: fineGrained).
The token `noname` has been saved to /root/.cache/huggingface/stored_tokens
Cannot authenticate through git-credential as no helper is defined on your machine.
You might have to re-authenticate when pushing to the Hugging Face Hub.
Run the following command in your terminal in case you want to set the 'store' credential helper as default.

git config --global credential.helper store

Read https://git-scm.com/book/en/v2/Git-Tools-Credential-Storage for more details.
Token has not been saved to git credential helper.
Your token has been saved to /root/.cache/huggingface/token
Login successful.
The current active token is: `noname`
```

```
pip install datasets huggingface_hub
```

Show hidden output

```
from datasets import load_dataset

# Replace with your username/dataset name
dataset = load_dataset("atulgupta002/banking_customer_service_query_intent")

# See available splits
print(dataset)

# Convert to DataFrame
df = dataset['train'].to_pandas()
df=df[["query","intent"]]
```

```
/usr/local/lib/python3.11/dist-packages/huggingface_hub/utils/_auth.py:94: UserWarning:
The secret `HF_TOKEN` does not exist in your Colab secrets.
To authenticate with the Hugging Face Hub, create a token in your settings tab (https://huggingface.co/settings/tokens), se
You will be able to reuse this secret in all of your notebooks.
Please note that authentication is recommended but still optional to access public models or datasets.
  warnings.warn(
DatasetDict({
    train: Dataset({
        features: ['Unnamed: 0', 'query', 'intent'],
        num_rows: 5000
    })
})
   Unnamed: 0                                              query  \
0           0  Could you please help me reset my account pass...
1           1               What company charged my account last?
2           2  How do I schedule an appointment to discuss lo...
3           3                          Which loans 2day suited me?
4           4  Do you have any updated timelines for my loan ...

            intent
0   password_reset
1  transaction_query
2      loan_inquiry
3      loan_inquiry
4      loan_inquiry
```

```
df=df[["query","intent"]]
```

```
df.sample(4)
```

	query	intent
1058	I would appreciate assistance in understanding...	credi_card_application
2103	Could you assist in determining if my account ...	loan_inquiry
1323	pls reset pass b4 i lose my mind	password_reset
679	how do i apply 4 biz credit?	loan_inquiry

```
df["intent"].value_counts()
```

intent	count
loan_inquiry	1600
fraud_report	1249
transaction_query	1210
balance_inquiry	323
credi_card_application	318
password_reset	300

dtype: int64

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import re
import nltk
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer
import warnings
warnings.filterwarnings('ignore')
nltk.download('stopwords')
nltk.download('punkt_tab')
nltk.download('wordnet')
```

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

```
Start coding or generate with AI.
```


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


```
df.shape
```

```
(5000, 2)
```

```
df.isnull().sum()
```

	0
query	0
intent	0

dtype: int64

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5000 entries, 0 to 4999
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   query   5000 non-null   object
 1   intent  5000 non-null   object
dtypes: object(2)
memory usage: 78.3+ KB
```

```
df.duplicated().sum()
```

```
np.int64(26)
```

```
df.drop_duplicates(inplace = True)
df.reset_index(drop=True, inplace=True)
```

```
df.duplicated().sum()
```

```
np.int64(0)
```

Text Preprocessing

```
df.sample(5)
```

	query	intent
1858	Was the most recent transaction completed with...	transaction_query
4035	ayo yall hacked me	fraud_report
4736	my pass expired, how do i reset?	password_reset
1743	What is the present financial status of my sav...	balance_inquiry
1882	May I ask for a thorough explanation of your l...	loan_inquiry

```
stop_words = set(stopwords.words('english'))
lemmatizer = WordNetLemmatizer()
```

```
list(stop_words)[:10]
```

```
['and',
 "she's",
 'is',
 "it's",
 "shouldn't",
 't',
 'if',
 "they'd",
 'had',
 "aren't"]
```

```
def preprocess_text(text):
    # 1. Lowercase
    text = text.lower()

    # 2. Remove punctuation and numbers
    text = re.sub(r'[^a-z\s]', '', text)  # keeps only letters and spaces

    # 3. Tokenize
    words = nltk.word_tokenize(text)

    # 4. Remove stopwords
    words = [word for word in words if word not in stop_words]

    # 5. Remove special characters
    words = [word for word in words if word.isalnum()==1]

    # 6. Lemmatize
    words = [lemmatizer.lemmatize(word) for word in words]
```

```
    # 7. Re-join words into a string
    return ' '.join(words)
```

```
for i in range(len(df)):
    df["query"][i]=preprocess_text(df["query"][i])
```

```
preprocess_text("ayo yall hacked me")
```

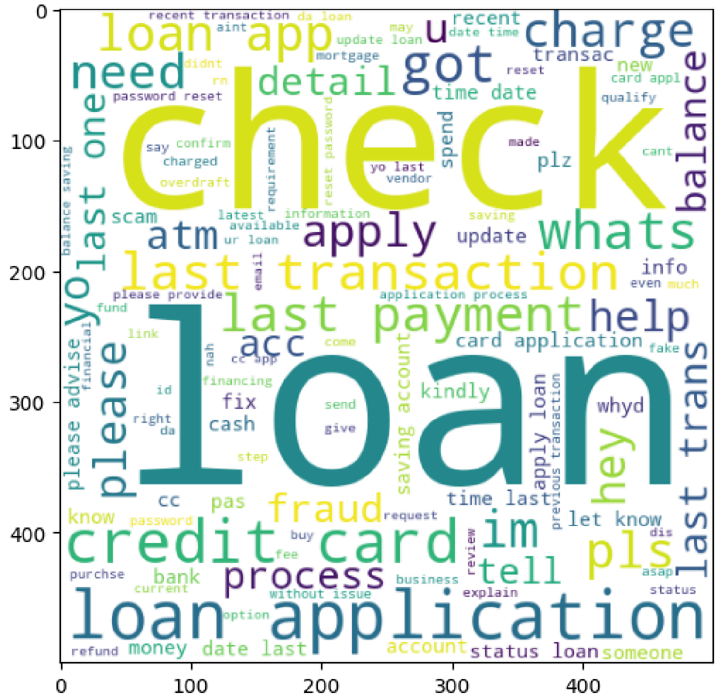
```
'ayo yall hacked'
```

```
from wordcloud import WordCloud
wc=WordCloud(width=500,height=500,min_font_size=10,background_color="white")
```

```
AI_wc=wc.generate(df["query"].str.cat(sep=" "))
```

```
plt.figure(figsize=(6,6))
plt.imshow(AI_wc)
```

```
<matplotlib.image.AxesImage at 0x7e111f7f65d0>
```



Feature Engineering

```
from sklearn.feature_extraction.text import TfidfVectorizer
tfidf=TfidfVectorizer(max_features=300, ngram_range=(1,3))
```

```
X=tfidf.fit_transform(df["query"]).toarray()
```

```
X.shape
```

```
(4974, 300)
```

```
from sentence_transformers import SentenceTransformer

model = SentenceTransformer('all-MiniLM-L6-v2')
embeddings = model.encode(df['query'].tolist())
```

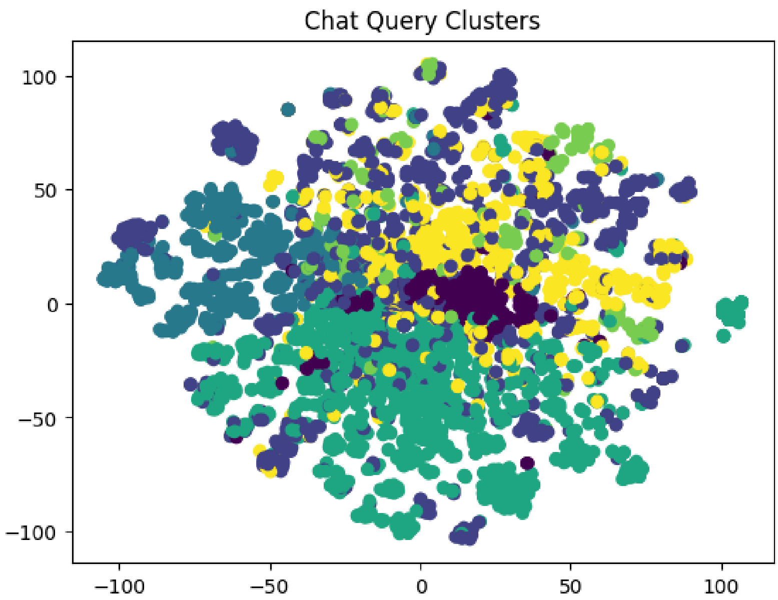
```
from sklearn.cluster import KMeans

kmeans = KMeans(n_clusters=3, random_state=42)
df['cluster'] = kmeans.fit_predict(embeddings)  # or 'embeddings' for SBERT
```

```
import hdbscan

clusterer = hdbscan.HDBSCAN(min_cluster_size=100)
df['cluster'] = clusterer.fit_predict(embeddings)
```

```
from sklearn.metrics import silhouette_score

# For vectorized data
score = silhouette_score(embeddings, df['cluster'])  # or 'embeddings'

print(f"Silhouette Score: {score:.2f}")  # Aim for >0.5
```

```
Silhouette Score: 0.09
```

```
from sklearn.manifold import TSNE
import matplotlib.pyplot as plt

tsne = TSNE(n_components=2, random_state=42)
vis_data = tsne.fit_transform(X)

plt.scatter(vis_data[:, 0], vis_data[:, 1], c=df['cluster'], cmap='viridis')
plt.title("Chat Query Clusters")
plt.show()
```




```
#pip install gensim
```

```
Collecting gensim
  Downloading gensim-4.3.3-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (8.1 kB)
Collecting numpy<2.0,>=1.18.5 (from gensim)
  Downloading numpy-1.26.4-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (61 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 61.0/61.0 kB 1.5 MB/s eta 0:00:00
Collecting scipy<1.14.0,>=1.7.0 (from gensim)
  Downloading scipy-1.13.1-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (60 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 60.6/60.6 kB 2.3 MB/s eta 0:00:00
Requirement already satisfied: smart-open>=1.8.1 in /usr/local/lib/python3.11/dist-packages (from gensim) (7.3.0.post1)
Requirement already satisfied: wrapt in /usr/local/lib/python3.11/dist-packages (from smart-open>=1.8.1->gensim) (1.17.3)
Downloading gensim-4.3.3-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (26.7 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 26.7/26.7 MB 31.5 MB/s eta 0:00:00
Downloading numpy-1.26.4-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (18.3 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 18.3/18.3 MB 39.6 MB/s eta 0:00:00
Downloading scipy-1.13.1-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (38.6 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 38.6/38.6 MB 8.6 MB/s eta 0:00:00
Installing collected packages: numpy, scipy, gensim
  Attempting uninstall: numpy
    Found existing installation: numpy 2.0.2
    Uninstalling numpy-2.0.2:
      Successfully uninstalled numpy-2.0.2
  Attempting uninstall: scipy
    Found existing installation: scipy 1.16.1
    Uninstalling scipy-1.16.1:
      Successfully uninstalled scipy-1.16.1
ERROR: pip's dependency resolver does not currently take into account all the packages that are installed. This behaviour is
opencv-contrib-python 4.12.0.88 requires numpy<2.3.0,>=2; python_version >= "3.9", but you have numpy 1.26.4 which is incomp
opencv-python 4.12.0.88 requires numpy<2.3.0,>=2; python_version >= "3.9", but you have numpy 1.26.4 which is incompatible.
thinc 8.3.6 requires numpy<3.0.0,>=2.0.0, but you have numpy 1.26.4 which is incompatible.
tsfresh 0.21.0 requires scipy>=1.14.0; python_version >= "3.10", but you have scipy 1.13.1 which is incompatible.
opencv-python-headless 4.12.0.88 requires numpy<2.3.0,>=2; python_version >= "3.9", but you have numpy 1.26.4 which is incon
Successfully installed gensim-4.3.3 numpy-1.26.4 scipy-1.13.1
```

```
from nltk.tokenize import word_tokenize
```

```
df
```

	query	intent
0	Could you please help me reset my account pass...	password_reset
1	What company charged my account last?	transaction_query
2	How do I schedule an appointment to discuss lo...	loan_inquiry
3	Which loans 2day suited me?	loan_inquiry
4	Do you have any updated timelines for my loan ...	loan_inquiry
...	...	...
4995	Was the transaction processed correctly?	transaction_query
4996	Apple Pay charges I didn't make—fraud!!	fraud_report
4997	Can you provide detailed steps on how to begin...	loan_inquiry
4998	what's the acc say? nervous chuckle	balance_inquiry
4999	why'd my transac get dunked on?	transaction_query

5000 rows × 2 columns

```
# Sample: Use your own dataframe
texts = df['query'].astype(str).apply(word_tokenize)
```

```
from gensim.models import Word2Vec

w2v_model = Word2Vec(sentences=texts,
                     vector_size=300,   # dimension of word embeddings
                     window=5,       # context window size
                     min_count=1,    # ignore rare words
                     sg=1)  # 1 = Skip-gram
```

```
w2v_model.wv['check']
```

```
        0.06067482, -0.1352418 , -0.09997616, -0.14746116, -0.06069662,
        0.12359828, -0.06053425,  0.03908334,  0.04258515, -0.09026598,
        0.12591277, -0.02877494, -0.0278694 , -0.06337349, -0.0172609 ,
       -0.11444492,  0.05766811,  0.1260676 , -0.05855938,  0.07825004,
        0.09201267,  0.04196916,  0.00710163, -0.12322395,  0.0706398 ,
        0.07292578,  0.08281413, -0.14102271,  0.09434362, -0.00499564,
        0.18015322,  0.06761307,  0.02011781,  0.07370505, -0.03664183,
       -0.16926664,  0.02060376, -0.07399998,  0.02030248, -0.09433614,
        0.09839907, -0.00811235,  0.09650765, -0.16685827, -0.08007952,
       -0.0062323 , -0.05684746,  0.0520971 , -0.08130912,  0.05558424,
        0.00258068, -0.1123684 ,  0.08319373, -0.19050603,  0.13416585,
       -0.1100857 , -0.06889507, -0.02476717,  0.18819547,  0.17773835,
       -0.03387957, -0.03506389,  0.01173825,  0.24117164,  0.03916995,
        0.08060808, -0.06059883,  0.0255092 ,  0.05085579,  0.12130864,
        0.14836572,  0.14520921, -0.03680674, -0.07277   ,  0.12838209,
       -0.05760913, -0.03922096, -0.00625797,  0.05260391,  0.00605842,
       -0.14735614,  0.03005108,  0.01942448, -0.06786844, -0.03596085,
       -0.1255352 , -0.06717985,  0.05114658,  0.07063763,  0.09512886,
        0.09014269,  0.01000162, -0.01996969,  0.10169414, -0.11314784,
        0.00635214,  0.08405048,  0.06247025, -0.05004987, -0.03075552,
        0.13315478,  0.03132945, -0.17855564, -0.08928069,  0.08867704,
        0.15002197,  0.22491665,  0.06825162, -0.142501  ,  0.11248266,
        0.05815551, -0.02215566, -0.05753402, -0.28935173, -0.1073303 ,
       -0.06780231, -0.06042719,  0.00780021,  0.04599118,  0.17170632,
       -0.15027045, -0.09992228, -0.01804444,  0.00607619, -0.13384832,
       -0.06030835, -0.14139393,  0.01912612,  0.03418608,  0.0011703 ,
        0.08823892, -0.21435651, -0.02992686,  0.01625939,  0.09992211,
        0.08922045, -0.02571659, -0.09545876,  0.02679662, -0.16964468,
        0.04496613,  0.06892719,  0.1362786 , -0.04137455,  0.25539932,
       -0.05382124,  0.00921551,  0.08995181,  0.10762193,  0.09171587,
        0.12721686,  0.0447987 , -0.14698721, -0.00759032,  0.07522756,
        0.04211498,  0.07298534, -0.11130349, -0.10016432, -0.05972821,
       -0.03809527,  0.2420915 ,  0.13757108, -0.05645635, -0.21128252,
        0.10414475,  0.09749319, -0.14725031,  0.07266144,  0.04379525,
       -0.10359757, -0.01876608, -0.12233123,  0.01370681, -0.06685211,
       -0.20239952, -0.05333187,  0.02983568, -0.04365081, -0.1136435 ,
       -0.09790797, -0.02202103, -0.03198808,  0.04528502, -0.03544129,
       -0.0322194 , -0.08054534, -0.17277747, -0.01997566, -0.04152491,
       -0.22001834, -0.13074744, -0.2677888 , -0.16012755, -0.05151961,
        0.04621465,  0.10063493, -0.1069029 , -0.03691696, -0.11325835,
       -0.04678367,  0.059754  , -0.02392766, -0.17169589,  0.11237822,
        0.01425483, -0.05468687, -0.09279235,  0.1527461 , -0.07922108,
        0.04868382,  0.06405529,  0.02595069,  0.03560789, -0.26492673,
        0.03376934, -0.02923445, -0.04274911, -0.03762026,  0.04080937,
       -0.11811687,  0.0562453 , -0.01940629,  0.10643192, -0.03013926,
        0.09165271, -0.02913047,  0.04234148, -0.03087159, -0.20593645,
       -0.02830286,  0.17284308, -0.0777771 , -0.24632865, -0.11375698,
        0.06621087,  0.16783036,  0.13950938, -0.262561  , -0.20037189,
        0.07623817, -0.00869674,  0.05241466, -0.08989535, -0.00063426,
        0.02333372,  0.05453069,  0.0844    ,  0.0197862 ,  0.0504894 ,
        0.15895598,  0.09731191,  0.01605176, -0.1981894 , -0.06992178,
       -0.01161425, -0.06605342, -0.07593374,  0.08435901,  0.08565657,
       -0.08661163, -0.09424403,  0.08685389,  0.00815491,  0.15802093,
        0.0327177 ,  0.24190313,  0.06422476,  0.00985391,  0.18354377,
        0.16362043,  0.04242164, -0.01480162,  0.12421991, -0.02877424],
      dtype=float32)
```

```
w2v_model.wv.most_similar('loan')
```

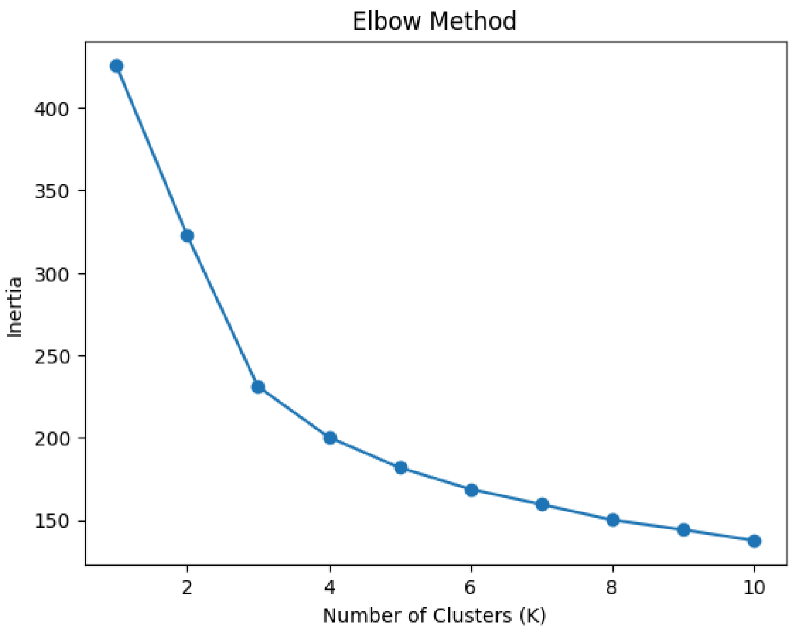
```
[('line', 0.9629036784172058),
 ('approval', 0.962518036365509),
 ('submit', 0.9612851142883301),
 ('credit', 0.9605515003204346),
 ('steps', 0.9584606289863586),
 ('on', 0.9576819539070129),
 ('personal', 0.9573211669921875),
 ('required', 0.9554418921470642),
 ('application', 0.954888641834259),
 ('regarding', 0.9509759545326233)]
```

```
import numpy as np
vocab_set = set(w2v_model.wv.index_to_key)
vector_size = w2v_model.vector_size
X_w2v = np.zeros((len(df), vector_size))
for i, text in enumerate(df['query']):
    vectors = [w2v_model.wv[word] for word in text if word in vocab_set]
    if vectors:
        X_w2v[i] = np.mean(vectors, axis=0)
```

```
X_w2v
```

```
array([[-0.0146107 ,  0.10901842, -0.04898347, ..., -0.05695681,
         0.1084467 , -0.06725479],
       [-0.01396885,  0.11685753, -0.0434003 , ..., -0.07728344,
         0.12547073, -0.05353657],
       [-0.015592  ,  0.1243165 , -0.05252567, ..., -0.06686812,
         0.11984246, -0.07384428],
       ...,
       [-0.0171927 ,  0.13332888, -0.03586521, ..., -0.08481096,
         0.12746266, -0.06543156],
       [-0.01875741,  0.11218677, -0.04241446, ..., -0.0511254 ,
         0.11027022, -0.06501191],
       [-0.01163105,  0.12245308, -0.06470449, ..., -0.06999706,
         0.12881836, -0.068672  ]])
```

```
from sklearn.cluster import KMeans
inertias = []
for k in range(1, 11):
    kmeans = KMeans(n_clusters=k, random_state=42)
    kmeans.fit(X_w2v)
    inertias.append(kmeans.inertia_)

import matplotlib.pyplot as plt
plt.plot(range(1, 11), inertias, marker='o')
plt.xlabel('Number of Clusters (K)')
plt.ylabel('Inertia')
plt.title('Elbow Method')
plt.show()
```




```
from sklearn.metrics import silhouette_score
silhouette_scores = []
for k in range(2, 11):
    kmeans = KMeans(n_clusters=k, random_state=42)
    clusters = kmeans.fit_predict(X_w2v)
    score = silhouette_score(X_w2v, clusters)
    silhouette_scores.append(score)
    print(f"K={k}, Silhouette Score={score:.2f}")
```

```
K=2, Silhouette Score=0.24
K=3, Silhouette Score=0.26
K=4, Silhouette Score=0.21
K=5, Silhouette Score=0.19
K=6, Silhouette Score=0.17
K=7, Silhouette Score=0.17
K=8, Silhouette Score=0.17
K=9, Silhouette Score=0.15
K=10, Silhouette Score=0.15
```

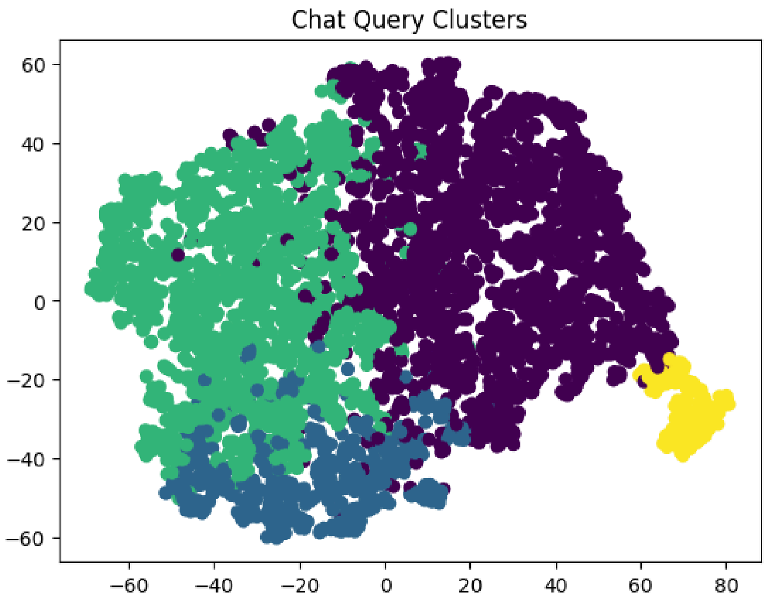
```
from sklearn.decomposition import TruncatedSVD
from sklearn.cluster import AgglomerativeClustering
```

```
# ==== 4. Agglomerative Clustering ====
n_clusters = 4  # you can change this
```

```
agg_clust = AgglomerativeClustering(n_clusters=n_clusters)
cluster_labels = agg_clust.fit_predict(X_w2v)

df['Agg_cluster'] = cluster_labels
from sklearn.metrics import silhouette_score
# ==== 5. Evaluation ====
sil_score = silhouette_score(X_w2v, cluster_labels)


print(f"Silhouette Score: {sil_score:.3f}")
```

```
Silhouette Score: 0.204
```

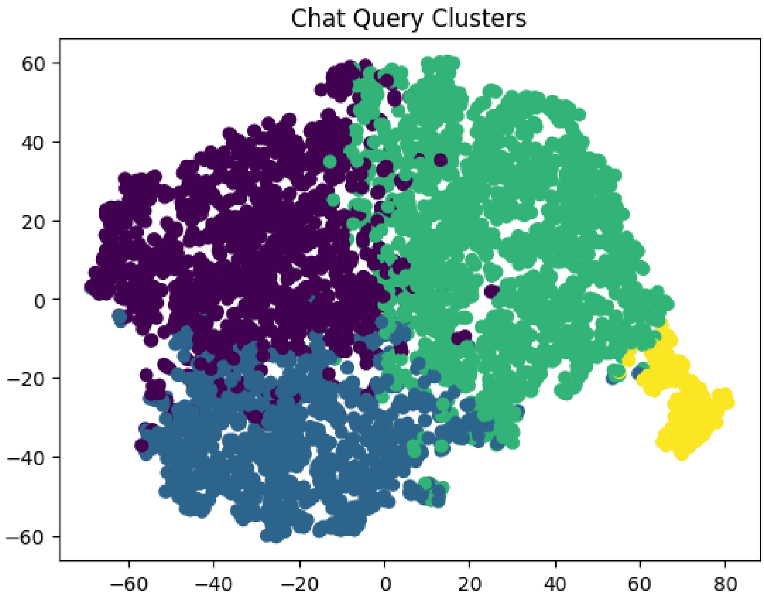
```
from sklearn.manifold import TSNE
import matplotlib.pyplot as plt

tsne = TSNE(n_components=2, random_state=42)
vis_data = tsne.fit_transform(X_w2v)

plt.scatter(vis_data[:, 0], vis_data[:, 1], c=cluster_labels, cmap='viridis')
plt.title("Chat Query Clusters")
plt.show()
```




```
# kmeans clustering
from sklearn.cluster import KMeans
kmeans = KMeans(n_clusters=4, random_state=42)
df['kmeans_cluster'] = kmeans.fit_predict(X_w2v)

sil_score = silhouette_score(X_w2v, df['kmeans_cluster'])


print(f"Silhouette Score: {sil_score:.3f}")
```

```
Silhouette Score: 0.215
```

```
from sklearn.manifold import TSNE
import matplotlib.pyplot as plt

tsne = TSNE(n_components=2, random_state=42)
vis_data = tsne.fit_transform(X_w2v)

plt.scatter(vis_data[:, 0], vis_data[:, 1], c=df['kmeans_cluster'], cmap='viridis')
plt.title("Chat Query Clusters")
plt.show()
```




```
# DBSCAN clustering
from sklearn.cluster import DBSCAN
dbscan = DBSCAN(eps=0.1, min_samples=5) # Adjusted parameters
df['dbscan_cluster'] = dbscan.fit_predict(X_w2v)

# Check the number of unique clusters found by DBSCAN
num_clusters = len(np.unique(df['dbscan_cluster']))

if num_clusters > 1:
  sil_score = silhouette_score(X_w2v, df['dbscan_cluster'])
  print(f"Silhouette Score: {sil_score:.3f}")
else:
  print("DBSCAN found only one cluster. Try adjusting eps and min_samples.")
```

```
Silhouette Score: -0.043
```

```
df
```

	query	intent	Agg_cluster	kmeans_cluster	dbscan_cluster
0	Could you please help me reset my account pass...	password_reset	1	1	0
1	What company charged my account last?	transaction_query	2	1	0
2	How do I schedule an appointment to discuss lo...	loan_inquiry	0	0	0
3	Which loans 2day suited me?	loan_inquiry	2	0	0
4	Do you have any updated timelines for my loan ...	loan_inquiry	2	0	0
...	...	...	...	...	...
4995	Was the transaction processed correctly?	transaction_query	0	2	0
4996	Apple Pay charges I didn't make—fraud!!	fraud_report	0	2	0
4997	Can you provide detailed steps on how to begin...	loan_inquiry	2	0	0
4998	what's the acc say? nervous chuckle	balance_inquiry	0	1	-1
4999	why'd my transac get dunked on?	transaction_query	0	2	0

5000 rows × 5 columns

```
import re
from sklearn.feature_extraction.text import TfidfVectorizer

# Function to clean text
def clean_text(text):
    text = text.lower()  # lowercase
    text = re.sub(r"[^a-zA-Z\s]", "", text)  # remove punctuation/numbers
    text = re.sub(r"\s+", " ", text).strip()  # remove extra spaces
    return text

# Apply cleaning
```

```
df["clean_query"] = df["query"].apply(clean_text)

# Vectorize using TF-IDF
vectorizer = TfidfVectorizer(stop_words="english", max_features=1000)
X_tfidf = vectorizer.fit_transform(df["clean_query"])

print("Shape of TF-IDF matrix:", X_tfidf.shape)
```

```
Shape of TF-IDF matrix: (5000, 1000)
```

```
from sklearn.decomposition import PCA

# Reduce dimensions for clustering
pca = PCA(n_components=50, random_state=42)  # keep top 50 components
X_pca = pca.fit_transform(X_tfidf.toarray())

print("Shape after PCA:", X_pca.shape)
```

```
Shape after PCA: (5000, 50)
```

```
!pip install sentence-transformers

from sentence_transformers import SentenceTransformer

# Load pre-trained SBERT model
model = SentenceTransformer('all-MiniLM-L6-v2')

# Encode sentences
X_embeddings = model.encode(df["clean_query"], show_progress_bar=True)

print("Shape of SBERT embeddings:", X_embeddings.shape)
```

```
Requirement already satisfied: sentence-transformers in /usr/local/lib/python3.11/dist-packages (5.1.0)
Requirement already satisfied: transformers<5.0.0,>=4.41.0 in /usr/local/lib/python3.11/dist-packages (from sentence-transfo
Requirement already satisfied: tqdm in /usr/local/lib/python3.11/dist-packages (from sentence-transformers) (4.67.1)
Requirement already satisfied: torch>=1.11.0 in /usr/local/lib/python3.11/dist-packages (from sentence-transformers) (2.6.0+
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.11/dist-packages (from sentence-transformers) (1.6.1)
Requirement already satisfied: scipy in /usr/local/lib/python3.11/dist-packages (from sentence-transformers) (1.13.1)
Requirement already satisfied: huggingface-hub>=0.20.0 in /usr/local/lib/python3.11/dist-packages (from sentence-transformer
Requirement already satisfied: Pillow in /usr/local/lib/python3.11/dist-packages (from sentence-transformers) (11.3.0)
Requirement already satisfied: typing_extensions>=4.5.0 in /usr/local/lib/python3.11/dist-packages (from sentence-transforme
Requirement already satisfied: filelock in /usr/local/lib/python3.11/dist-packages (from huggingface-hub>=0.20.0->sentence-t
Requirement already satisfied: fsspec>=2023.5.0 in /usr/local/lib/python3.11/dist-packages (from huggingface-hub>=0.20.0->se
Requirement already satisfied: packaging>=20.9 in /usr/local/lib/python3.11/dist-packages (from huggingface-hub>=0.20.0->ser
Requirement already satisfied: pyyaml>=5.1 in /usr/local/lib/python3.11/dist-packages (from huggingface-hub>=0.20.0->senten
Requirement already satisfied: requests in /usr/local/lib/python3.11/dist-packages (from huggingface-hub>=0.20.0->sentence-t
Requirement already satisfied: hf-xet<2.0.0,>=1.1.3 in /usr/local/lib/python3.11/dist-packages (from huggingface-hub>=0.20.0
Requirement already satisfied: networkx in /usr/local/lib/python3.11/dist-packages (from torch>=1.11.0->sentence-transformer
Requirement already satisfied: jinja2 in /usr/local/lib/python3.11/dist-packages (from torch>=1.11.0->sentence-transformers)
Collecting nvidia-cuda-nvrtc-cu12==12.4.127 (from torch>=1.11.0->sentence-transformers)
  Downloading nvidia_cuda_nvrtc_cu12-12.4.127-py3-none-manylinux2014_x86_64.whl.metadata (1.5 kB)
Collecting nvidia-cuda-runtime-cu12==12.4.127 (from torch>=1.11.0->sentence-transformers)
  Downloading nvidia_cuda_runtime_cu12-12.4.127-py3-none-manylinux2014_x86_64.whl.metadata (1.5 kB)
Collecting nvidia-cuda-cupti-cu12==12.4.127 (from torch>=1.11.0->sentence-transformers)
  Downloading nvidia_cuda_cupti_cu12-12.4.127-py3-none-manylinux2014_x86_64.whl.metadata (1.6 kB)
Collecting nvidia-cudnn-cu12==9.1.0.70 (from torch>=1.11.0->sentence-transformers)
  Downloading nvidia_cudnn_cu12-9.1.0.70-py3-none-manylinux2014_x86_64.whl.metadata (1.6 kB)
Collecting nvidia-cublas-cu12==12.4.5.8 (from torch>=1.11.0->sentence-transformers)
  Downloading nvidia_cublas_cu12-12.4.5.8-py3-none-manylinux2014_x86_64.whl.metadata (1.5 kB)
Collecting nvidia-cufft-cu12==11.2.1.3 (from torch>=1.11.0->sentence-transformers)
  Downloading nvidia_cufft_cu12-11.2.1.3-py3-none-manylinux2014_x86_64.whl.metadata (1.5 kB)
Collecting nvidia-curand-cu12==10.3.5.147 (from torch>=1.11.0->sentence-transformers)
  Downloading nvidia_curand_cu12-10.3.5.147-py3-none-manylinux2014_x86_64.whl.metadata (1.5 kB)
Collecting nvidia-cusolver-cu12==11.6.1.9 (from torch>=1.11.0->sentence-transformers)
  Downloading nvidia_cusolver_cu12-11.6.1.9-py3-none-manylinux2014_x86_64.whl.metadata (1.6 kB)
Collecting nvidia-cusparse-cu12==12.3.1.170 (from torch>=1.11.0->sentence-transformers)
  Downloading nvidia_cusparse_cu12-12.3.1.170-py3-none-manylinux2014_x86_64.whl.metadata (1.6 kB)
Requirement already satisfied: nvidia-cusparselt-cu12==0.6.2 in /usr/local/lib/python3.11/dist-packages (from torch>=1.11.0-
Collecting nvidia-nccl-cu12==2.21.5 (from torch>=1.11.0->sentence-transformers)
  Downloading nvidia_nccl_cu12-2.21.5-py3-none-manylinux2014_x86_64.whl.metadata (1.8 kB)
Requirement already satisfied: nvidia-nvtx-cu12==12.4.127 in /usr/local/lib/python3.11/dist-packages (from torch>=1.11.0->se
Collecting nvidia-nvjitlink-cu12==12.4.127 (from torch>=1.11.0->sentence-transformers)
  Downloading nvidia_nvjitlink_cu12-12.4.127-py3-none-manylinux2014_x86_64.whl.metadata (1.5 kB)
Requirement already satisfied: triton==3.2.0 in /usr/local/lib/python3.11/dist-packages (from torch>=1.11.0->sentence-transf
Requirement already satisfied: sympy==1.13.1 in /usr/local/lib/python3.11/dist-packages (from torch>=1.11.0->sentence-transf
Requirement already satisfied: mpmath<1.4,>=1.1.0 in /usr/local/lib/python3.11/dist-packages (from sympy==1.13.1->torch>=1.1
Requirement already satisfied: numpy>=1.17 in /usr/local/lib/python3.11/dist-packages (from transformers<5.0.0,>=4.41.0->ser
Requirement already satisfied: regex!=2019.12.17 in /usr/local/lib/python3.11/dist-packages (from transformers<5.0.0,>=4.41
Requirement already satisfied: tokenizers<0.22,>=0.21 in /usr/local/lib/python3.11/dist-packages (from transformers<5.0.0,>=
Requirement already satisfied: safetensors>=0.4.3 in /usr/local/lib/python3.11/dist-packages (from transformers<5.0.0,>=4.41
Requirement already satisfied: joblib>=1.2.0 in /usr/local/lib/python3.11/dist-packages (from scikit-learn->sentence-transfo
Requirement already satisfied: threadpoolctl>=3.1.0 in /usr/local/lib/python3.11/dist-packages (from scikit-learn->sentence-
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.11/dist-packages (from jinja2->torch>=1.11.0->sente
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.11/dist-packages (from requests->huggingfa
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.11/dist-packages (from requests->huggingface-hub>=0.20
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.11/dist-packages (from requests->huggingface-hub
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.11/dist-packages (from requests->huggingface-hub
Downloading nvidia_cublas_cu12-12.4.5.8-py3-none-manylinux2014_x86_64.whl (363.4 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 363.4/363.4 MB 1.3 MB/s eta 0:00:00
Downloading nvidia_cuda_cupti_cu12-12.4.127-py3-none-manylinux2014_x86_64.whl (13.8 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 13.8/13.8 MB 98.8 MB/s eta 0:00:00
Downloading nvidia_cuda_nvrtc_cu12-12.4.127-py3-none-manylinux2014_x86_64.whl (24.6 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 24.6/24.6 MB 78.5 MB/s eta 0:00:00
Downloading nvidia_cuda_runtime_cu12-12.4.127-py3-none-manylinux2014_x86_64.whl (883 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 883.7/883.7 kB 51.5 MB/s eta 0:00:00
Downloading nvidia_cudnn_cu12-9.1.0.70-py3-none-manylinux2014_x86_64.whl (664.8 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 664.8/664.8 MB 966.7 kB/s eta 0:00:00
Downloading nvidia_cufft_cu12-11.2.1.3-py3-none-manylinux2014_x86_64.whl (211.5 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 211.5/211.5 MB 3.2 MB/s eta 0:00:00
Downloading nvidia_curand_cu12-10.3.5.147-py3-none-manylinux2014_x86_64.whl (56.3 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 56.3/56.3 MB 9.6 MB/s eta 0:00:00
Downloading nvidia_cusolver_cu12-11.6.1.9-py3-none-manylinux2014_x86_64.whl (127.9 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 127.9/127.9 MB 4.5 MB/s eta 0:00:00
Downloading nvidia_cusparse_cu12-12.3.1.170-py3-none-manylinux2014_x86_64.whl (207.5 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 207.5/207.5 MB 4.4 MB/s eta 0:00:00
Downloading nvidia_nccl_cu12-2.21.5-py3-none-manylinux2014_x86_64.whl (188.7 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 188.7/188.7 MB 4.8 MB/s eta 0:00:00
Downloading nvidia_nvjitlink_cu12-12.4.127-py3-none-manylinux2014_x86_64.whl (21.1 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 21.1/21.1 MB 88.7 MB/s eta 0:00:00
Installing collected packages: nvidia-nvjitlink-cu12, nvidia-nccl-cu12, nvidia-curand-cu12, nvidia-cufft-cu12, nvidia-cuda-r
  Attempting uninstall: nvidia-nvjitlink-cu12
    Found existing installation: nvidia-nvjitlink-cu12 12.5.82
    Uninstalling nvidia-nvjitlink-cu12-12.5.82:
      Successfully uninstalled nvidia-nvjitlink-cu12-12.5.82
  Attempting uninstall: nvidia-nccl-cu12
    Found existing installation: nvidia-nccl-cu12 2.23.4
    Uninstalling nvidia-nccl-cu12-2.23.4:
```

```
      Successfully uninstalled nvidia-nccl-cu12-2.23.4
  Attempting uninstall: nvidia-curand-cu12
    Found existing installation: nvidia-curand-cu12 10.3.6.82
    Uninstalling nvidia-curand-cu12-10.3.6.82:
      Successfully uninstalled nvidia-curand-cu12-10.3.6.82
  Attempting uninstall: nvidia-cufft-cu12
    Found existing installation: nvidia-cufft-cu12 11.2.3.61
    Uninstalling nvidia-cufft-cu12-11.2.3.61:
      Successfully uninstalled nvidia-cufft-cu12-11.2.3.61
  Attempting uninstall: nvidia-cuda-runtime-cu12
    Found existing installation: nvidia-cuda-runtime-cu12 12.5.82
    Uninstalling nvidia-cuda-runtime-cu12-12.5.82:
      Successfully uninstalled nvidia-cuda-runtime-cu12-12.5.82
  Attempting uninstall: nvidia-cuda-nvrtc-cu12
    Found existing installation: nvidia-cuda-nvrtc-cu12 12.5.82
    Uninstalling nvidia-cuda-nvrtc-cu12-12.5.82:
      Successfully uninstalled nvidia-cuda-nvrtc-cu12-12.5.82
  Attempting uninstall: nvidia-cuda-cupti-cu12
    Found existing installation: nvidia-cuda-cupti-cu12 12.5.82
    Uninstalling nvidia-cuda-cupti-cu12-12.5.82:
      Successfully uninstalled nvidia-cuda-cupti-cu12-12.5.82
  Attempting uninstall: nvidia-cublas-cu12
    Found existing installation: nvidia-cublas-cu12 12.5.3.2
    Uninstalling nvidia-cublas-cu12-12.5.3.2:
      Successfully uninstalled nvidia-cublas-cu12-12.5.3.2
  Attempting uninstall: nvidia-cusparse-cu12
    Found existing installation: nvidia-cusparse-cu12 12.5.1.3
    Uninstalling nvidia-cusparse-cu12-12.5.1.3:
      Successfully uninstalled nvidia-cusparse-cu12-12.5.1.3
  Attempting uninstall: nvidia-cudnn-cu12
    Found existing installation: nvidia-cudnn-cu12 9.3.0.75
    Uninstalling nvidia-cudnn-cu12-9.3.0.75:
      Successfully uninstalled nvidia-cudnn-cu12-9.3.0.75
  Attempting uninstall: nvidia-cusolver-cu12
    Found existing installation: nvidia-cusolver-cu12 11.6.3.83
    Uninstalling nvidia-cusolver-cu12-11.6.3.83:
      Successfully uninstalled nvidia-cusolver-cu12-11.6.3.83
Successfully installed nvidia-cublas-cu12-12.4.5.8 nvidia-cuda-cupti-cu12-12.4.127 nvidia-cuda-nvrtc-cu12-12.4.127 nvidia-cu
```

Batches: 100% 157/157 [00:32<00:00, 8.68it/s]

```
Shape of SBERT embeddings: (5000, 384)
```

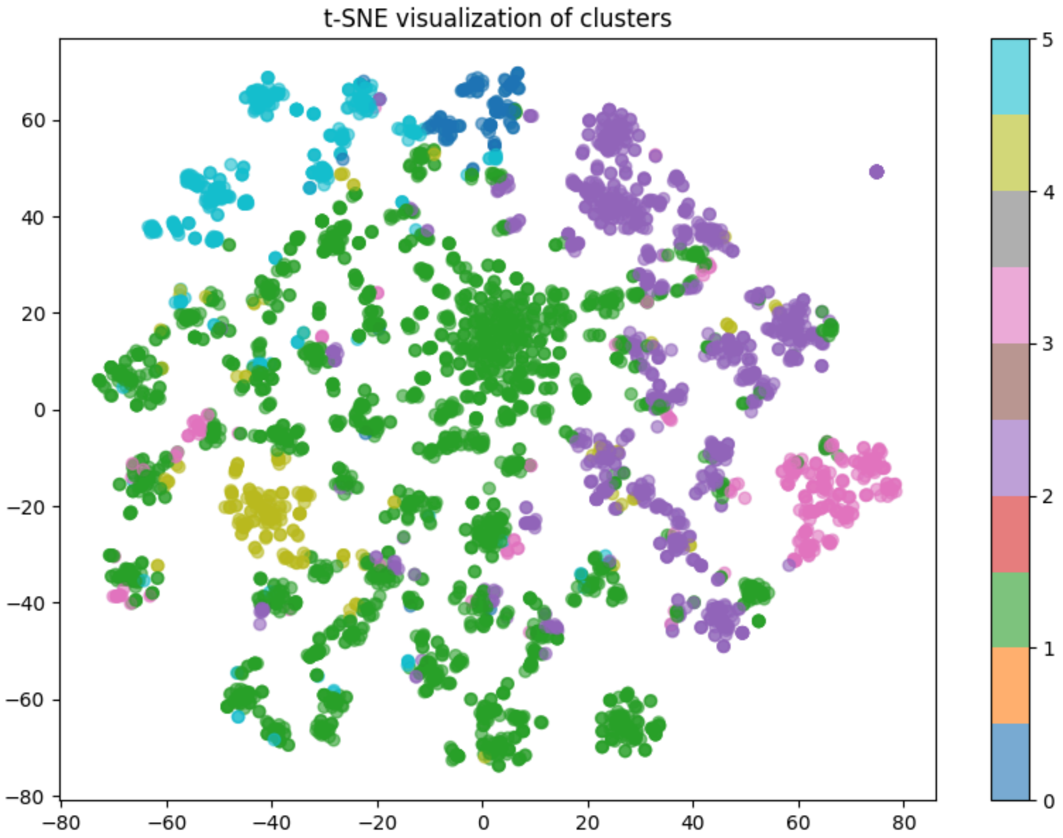
```
from sklearn.cluster import KMeans
import numpy as np

# Choose embeddings (pick X_pca or X_embeddings)
#X = X_embeddings  # if using SBERT
X = X_pca       # if using TF-IDF + PCA

# Define KMeans
kmeans = KMeans(n_clusters=6, random_state=42, n_init=10)
clusters = kmeans.fit_predict(X)

# Add cluster labels to dataframe
df["cluster"] = clusters

# Check some results
df[["query", "intent", "cluster"]].sample(20)
```

	query	intent	cluster
2335	Something looks off with my last few transacti...	fraud_report	1
3594	i never used atm that night	fraud_report	1
62	What vendor processed my most recent charge?	transaction_query	1
67	too many charges poppin up	fraud_report	1
2364	I have forgotten my password; please help me r...	password_reset	1
3467	Did the app break? WHY CAN’T I SEE MY SAVINGS?!	balance_inquiry	5
1386	Was the last payment made correctly?	transaction_query	1
3564	business lending types?	loan_inquiry	1
1845	whut loan solutions do u have?	loan_inquiry	2
3141	Was the previous transaction done without issues?	transaction_query	5
3083	I’m having a crisis—SAVINGS BALANCE IMMEDIATELY!	balance_inquiry	1
306	Did the most recent transaction finalize witho...	transaction_query	5
318	wuts dis brand lol	fraud_report	1
570	yo, any sign on my loan app progressin’?	loan_inquiry	2
4122	I’d like to learn about in-branch application ...	credi_card_application	1
4594	they said my data is leaking, pay up or else	fraud_report	1
827	Has my loan request been forwarded to the appr...	loan_inquiry	2
1081	transacshun? panics in millennial	transaction_query	1
3049	overdraft woes galore	fraud_report	1
4232	Please guide me through the password reset pro...	password_reset	1

```
from sklearn.metrics import adjusted_rand_score, normalized_mutual_info_score

# True labels (ground truth)
y_true = df["intent"]

# Predicted cluster labels
y_pred = df["cluster"]

# Adjusted Rand Index (ARI): how well clustering matches true labels (0 = random, 1 = perfect)
ari = adjusted_rand_score(y_true, y_pred)

# Normalized Mutual Information (NMI): measures shared information (0 to 1)
nmi = normalized_mutual_info_score(y_true, y_pred)

print("Adjusted Rand Index (ARI):", ari)
print("Normalized Mutual Information (NMI):", nmi)
sil_score = silhouette_score(X, df['cluster'])


print(f"Silhouette Score: {sil_score:.3f}")
```

```
Adjusted Rand Index (ARI): 0.2206192099643765
Normalized Mutual Information (NMI): 0.3840869184207783
Silhouette Score: 0.103
```

```
from sklearn.manifold import TSNE
import matplotlib.pyplot as plt

# Reduce embeddings to 2D for visualization
tsne = TSNE(n_components=2, random_state=42, perplexity=30)
X_2d = tsne.fit_transform(X)

# Plot clusters
plt.figure(figsize=(10,7))
plt.scatter(X_2d[:,0], X_2d[:,1], c=df["cluster"], cmap="tab10", alpha=0.6)
plt.colorbar()
plt.title("t-SNE visualization of clusters")
plt.show()
```




```
plt.figure(figsize=(10,7))
plt.scatter(X_2d[:,0], X_2d[:,1], c=df["intent"].astype('category').cat.codes, cmap="tab10", alpha=0.6)
plt.colorbar()
plt.title("t-SNE visualization by True Intents")
plt.show()
```


```
from sentence_transformers import SentenceTransformer
from sklearn.cluster import KMeans
```

```
# Load a pre-trained sentence embedding model
model = SentenceTransformer("all-MiniLM-L6-v2")

# Encode all queries
embeddings = model.encode(df["query"].tolist(), show_progress_bar=True)

# Cluster
kmeans = KMeans(n_clusters=4, random_state=42)
df["cluster"] = kmeans.fit_predict(embeddings)
```

Batches: 100% 157/157 [00:37<00:00, 7.24it/s]

```
from hdbscan import HDBSCAN

clusterer = HDBSCAN(min_cluster_size=10, metric='euclidean')
df["cluster"] = clusterer.fit_predict(embeddings)
```

```
/usr/local/lib/python3.11/dist-packages/sklearn/utils/deprecation.py:151: FutureWarning: 'force_all_finite' was renamed to
  warnings.warn(
/usr/local/lib/python3.11/dist-packages/sklearn/utils/deprecation.py:151: FutureWarning: 'force_all_finite' was renamed to
  warnings.warn(
```

```
from sklearn.metrics import adjusted_rand_score, normalized_mutual_info_score

ari = adjusted_rand_score(df["intent"], df["cluster"])
nmi = normalized_mutual_info_score(df["intent"], df["cluster"])
print("ARI:", ari)
print("NMI:", nmi)
```

```
ARI: 0.21203621492640584
NMI: 0.4686243542822578
```

UMAP & T-SNE

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import re
import nltk
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer
import warnings
warnings.filterwarnings('ignore')
nltk.download('all')
```

Show hidden output

```
from datasets import load_dataset

# Replace with your username/dataset name
dataset = load_dataset("atulgupta002/banking_customer_service_query_intent")

# See available splits
print(dataset)

# Convert to DataFrame
df = dataset['train'].to_pandas()
df=df[["query","intent"]]
```

README.md: 100% 312/312 [00:00<00:00, 34.1kB/s]

(…)ervice_intent_classification_dataset.csv: 326k/? [00:00<00:00, 16.6MB/s]

Generating train split: 100% 5000/5000 [00:00<00:00, 108007.09 examples/s]

```
DatasetDict({
    train: Dataset({
        features: ['Unnamed: 0', 'query', 'intent'],
        num_rows: 5000
    })
})
```

```
df
```

	query	intent
0	Could you please help me reset my account pass...	password_reset
1	What company charged my account last?	transaction_query
2	How do I schedule an appointment to discuss lo...	loan_inquiry
3	Which loans 2day suited me?	loan_inquiry
4	Do you have any updated timelines for my loan ...	loan_inquiry
...	...	...
4995	Was the transaction processed correctly?	transaction_query
4996	Apple Pay charges I didn’t make—fraud!!	fraud_report
4997	Can you provide detailed steps on how to begin...	loan_inquiry
4998	what’s the acc say? nervous chuckle	balance_inquiry
4999	why’d my transac get dunked on?	transaction_query

5000 rows × 2 columns

```
df.columns=["t","i"]
```

```
df["n_c"]=df["t"].apply(len)
```

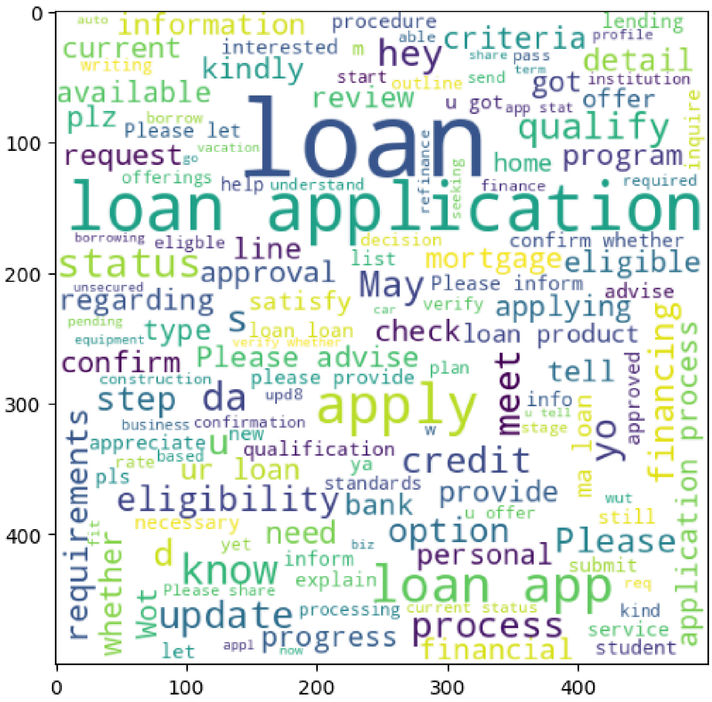
```
df["n_w"]=df["t"].apply(lambda  x:len(nltk.word_tokenize(x)))
```

```
df["n_s"]=df["t"].apply(lambda  x:len(nltk.sent_tokenize(x)))
```

```
from wordcloud import WordCloud
wc=WordCloud(width=500,height=500,min_font_size=10,background_color="white")
```

```
AI_wc=wc.generate(df[df["i"]=="loan_inquiry"]["t"].str.cat(sep=" "))
plt.figure(figsize=(6,6))
plt.imshow(AI_wc)
```

```
<matplotlib.image.AxesImage at 0x7a7dc8422f90>
```

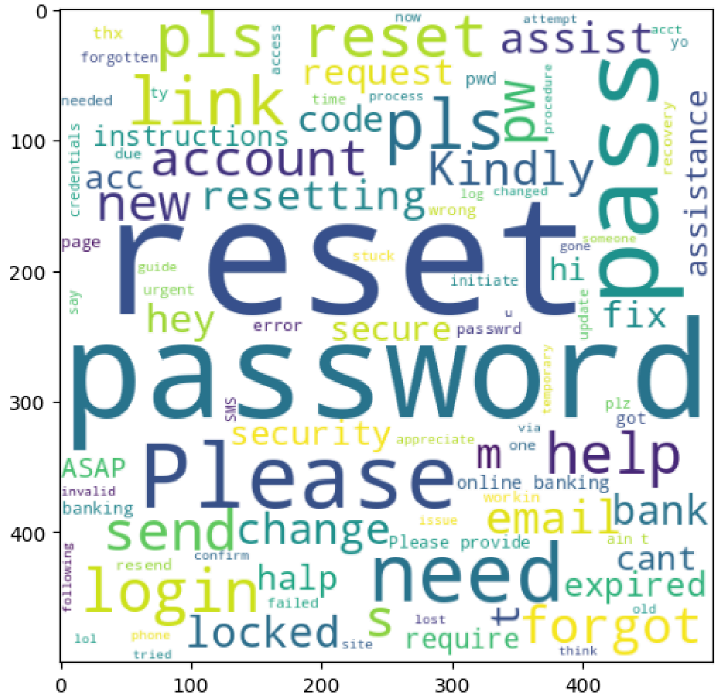



```
df[df["i"]=="loan_inquiry"][["n_c","n_w","n_s"]].describe()
```

	n_c	n_w	n_s
count	1600.000000	1600.000000	1600.000000
mean	47.361250	10.124375	1.011875
std	21.274983	3.323524	0.108357
min	12.000000	3.000000	1.000000
25%	28.000000	7.000000	1.000000
50%	44.000000	10.000000	1.000000
75%	64.250000	13.000000	1.000000
max	112.000000	20.000000	2.000000

```
AI_wc=wc.generate(df[df["i"]=="password_reset"]["t"].str.cat(sep=" "))
plt.figure(figsize=(6,6))
plt.imshow(AI_wc)
```

```
<matplotlib.image.AxesImage at 0x7a7dc5e0a490>
```




```
df[df["i"]=="password_reset"][["n_c","n_w","n_s"]].describe()
```

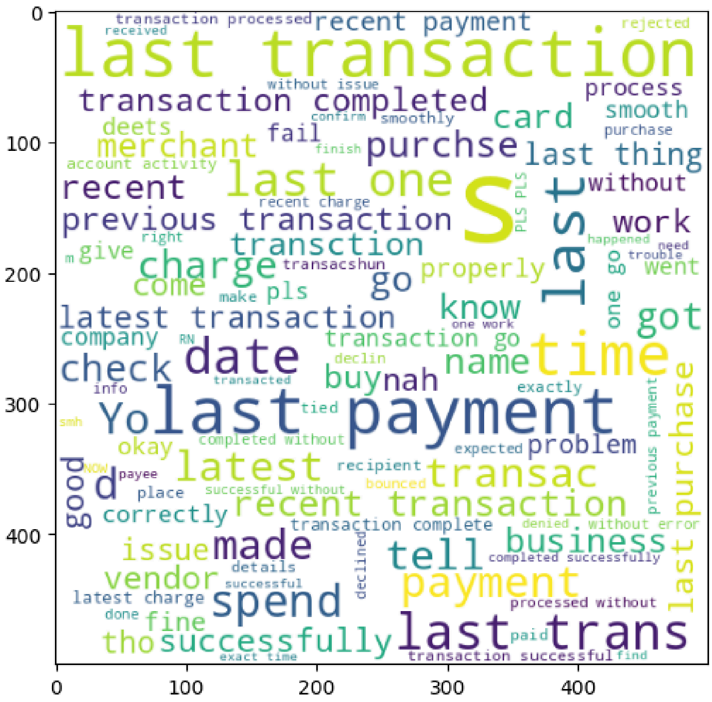
	n_c	n_w	n_s
count	300.000000	300.000000	300.000000
mean	40.050000	8.613333	1.233333
std	11.561418	2.258501	0.461446
min	20.000000	4.000000	1.000000
25%	31.000000	7.000000	1.000000
50%	37.000000	9.000000	1.000000
75%	49.000000	10.000000	1.000000
max	76.000000	14.000000	3.000000

```
import tensorflow as tf

# Check available GPUs
gpus = tf.config.list_physical_devices('GPU')
if gpus:
    try:
        # Set memory growth (prevents TensorFlow from grabbing all GPU memory at once)
        for gpu in gpus:
            tf.config.experimental.set_memory_growth(gpu, True)
        print(f"TensorFlow is using GPU: {gpus}")
    except RuntimeError as e:
        print(e)
else:
    print("No GPU found. TensorFlow is running on CPU.")
```

```
TensorFlow is using GPU: [PhysicalDevice(name='/physical_device:GPU:0', device_type='GPU')]
```

```
AI_wc=wc.generate(df[df["i"]=="transaction_query"]["t"].str.cat(sep=" "))
plt.figure(figsize=(6,6))
plt.imshow(AI_wc)
```

```
<matplotlib.image.AxesImage at 0x7a7d65317ad0>
```

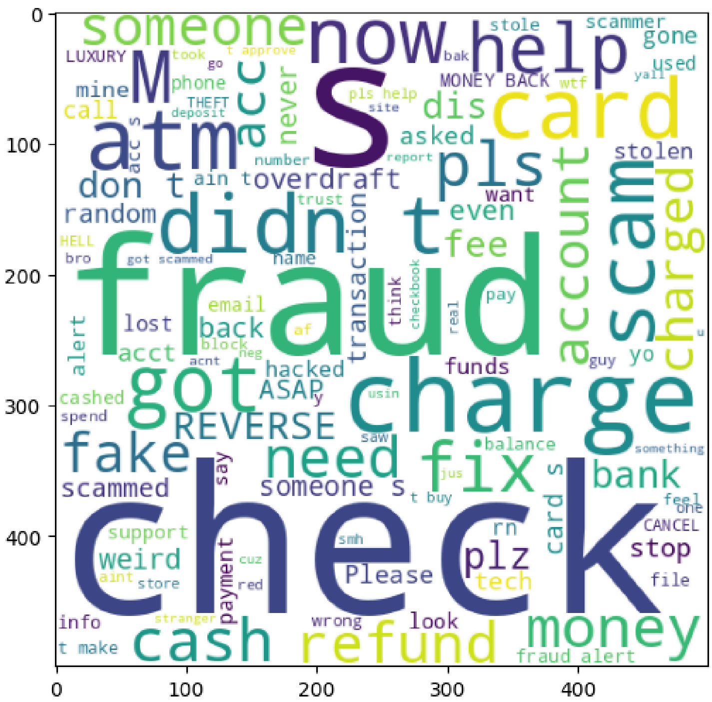



```
df[df["i"]=="transaction_query"][["n_c","n_w","n_s"]].describe()
```

	n_c	n_w	n_s
count	1210.000000	1210.000000	1210.000000
mean	40.805785	8.990083	1.090083
std	10.304919	2.380629	0.286419
min	16.000000	3.000000	1.000000
25%	32.000000	7.000000	1.000000
50%	40.000000	9.000000	1.000000
75%	49.000000	10.000000	1.000000
max	68.000000	17.000000	2.000000

```
# fraud_report
AI_wc=wc.generate(df[df["i"]=="fraud_report"]["t"].str.cat(sep=" "))
plt.figure(figsize=(6,6))
plt.imshow(AI_wc)
```

```
<matplotlib.image.AxesImage at 0x7a7d44d08d10>
```

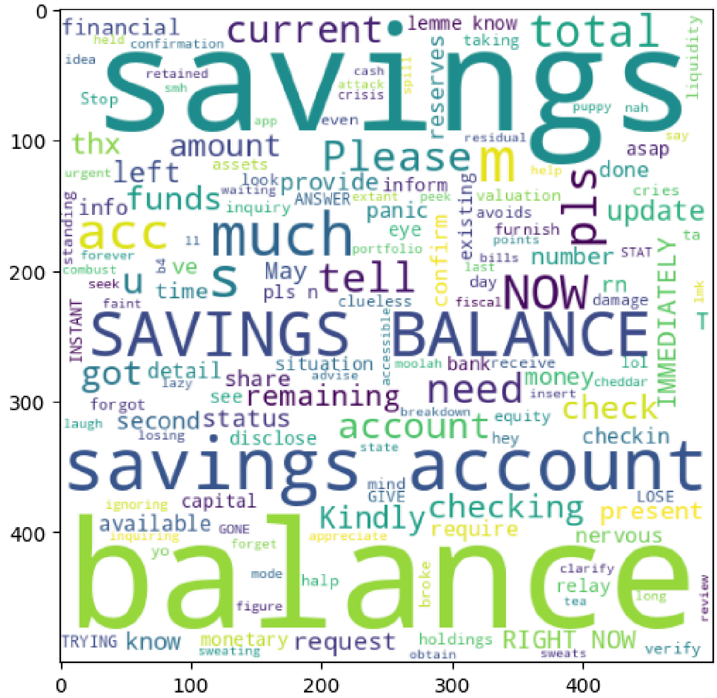



```
df[df["i"]=="fraud_report"][["n_c","n_w","n_s"]].describe()
```

	n_c	n_w	n_s
count	1249.000000	1249.000000	1249.000000
mean	37.662930	9.349880	1.495596
std	14.487204	4.348418	0.673577
min	9.000000	2.000000	1.000000
25%	25.000000	6.000000	1.000000
50%	38.000000	9.000000	1.000000
75%	46.000000	12.000000	2.000000
max	91.000000	25.000000	4.000000

```
AI_wc=wc.generate(df[df["i"]=="balance_inquiry"]["t"].str.cat(sep=" "))
plt.figure(figsize=(6,6))
plt.imshow(AI_wc)
```

```
<matplotlib.image.AxesImage at 0x7a7d44d7bf50>
```

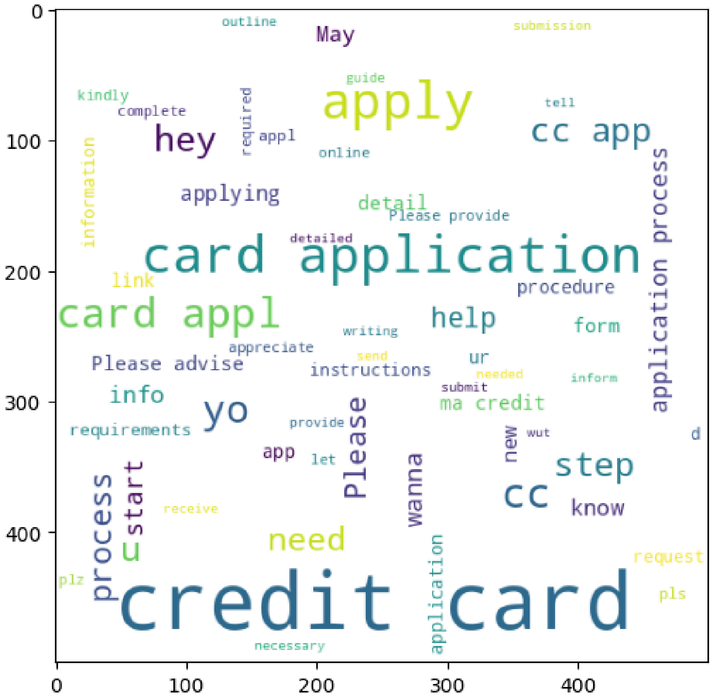



```
df[df["i"]=="balance_inquiry"][["n_c","n_w","n_s"]].describe()
```

	n_c	n_w	n_s
count	323.000000	323.000000	323.000000
mean	43.894737	9.674923	1.448916
std	8.727486	2.121223	0.695833
min	7.000000	1.000000	1.000000
25%	37.000000	9.000000	1.000000
50%	46.000000	10.000000	1.000000
75%	50.000000	11.000000	2.000000
max	74.000000	16.000000	7.000000

```
AI_wc=wc.generate(df[df["i"]=="credi_card_application"]["t"].str.cat(sep=" "))
plt.figure(figsize=(6,6))
plt.imshow(AI_wc)
```

```
<matplotlib.image.AxesImage at 0x7a7d44d6ccd0>
```




```
df[df["i"]=="credi_card_application"][["n_c","n_w","n_s"]].describe()
```

	n_c	n_w	n_s
count	318.000000	318.000000	318.000000
mean	52.924528	11.327044	1.022013
std	22.129971	3.370866	0.146956
min	11.000000	4.000000	1.000000
25%	31.000000	8.000000	1.000000
50%	51.000000	12.000000	1.000000
75%	72.000000	14.000000	1.000000
max	107.000000	19.000000	2.000000

```
df
```

	t	i	n_c	n_w	n_s
0	Could you please help me reset my account pass...	password_reset	51	10	1
1	What company charged my account last?	transaction_query	37	7	1
2	How do I schedule an appointment to discuss lo...	loan_inquiry	67	13	1
3	Which loans 2day suited me?	loan_inquiry	27	6	1
4	Do you have any updated timelines for my loan ...	loan_inquiry	55	11	1
...	...	...	...	...	...
4995	Was the transaction processed correctly?	transaction_query	40	6	1
4996	Apple Pay charges I didn’t make—fraud!!	fraud_report	39	10	2
4997	Can you provide detailed steps on how to begin...	loan_inquiry	66	13	1
4998	what’s the acc say? nervous chuckle	balance_inquiry	35	9	2
4999	why’d my transac get dunked on?	transaction_query	31	9	1

5000 rows × 5 columns

```
from sentence_transformers import SentenceTransformer
# Load a pre-trained sentence embedding model
model = SentenceTransformer("all-MiniLM-L6-v2")
```

```
# Encode all queries
embeddings = model.encode(df["t"].tolist(), show_progress_bar=True)
```

modules.json: 100% 349/349 [00:00<00:00, 24.3kB/s]

config_sentence_transformers.json: 100% 116/116 [00:00<00:00, 6.46kB/s]

README.md: 10.5k/? [00:00<00:00, 595kB/s]

sentence_bert_config.json: 100% 53.0/53.0 [00:00<00:00, 3.56kB/s]

config.json: 100% 612/612 [00:00<00:00, 39.3kB/s]

model.safetensors: 100% 90.9M/90.9M [00:01<00:00, 80.1MB/s]

tokenizer_config.json: 100% 350/350 [00:00<00:00, 9.62kB/s]

vocab.txt: 232k/? [00:00<00:00, 5.55MB/s]

tokenizer.json: 466k/? [00:00<00:00, 22.7MB/s]

special_tokens_map.json: 100% 112/112 [00:00<00:00, 8.01kB/s]

config.json: 100% 190/190 [00:00<00:00, 4.87kB/s]

Batches: 100% 157/157 [00:02<00:00, 140.24it/s]

```
embeddings
```

```
array([[-0.03671908, -0.07597653, -0.01249438, ...,  0.0780434 ,
        -0.02108414, -0.10642095],
       [ 0.00080891, -0.01177414, -0.02466845, ..., -0.11604776,
         0.04516524, -0.01950026],
       [ 0.0100682 , -0.00434602, -0.00085265, ...,  0.02112929,
        -0.09129895, -0.06690539],
       ...,
       [ 0.0392577 ,  0.03649173, -0.0212892 , ...,  0.08284812,
        -0.02740316, -0.06003959],
       [-0.08713641, -0.00544636,  0.02045082, ...,  0.06934794,
         0.02031212, -0.03548901],
       [ 0.01072836, -0.02365415,  0.05852266, ..., -0.01786028,
        -0.05420143,  0.01089964]], dtype=float32)
```

```
X_new=pd.concat([pd.DataFrame(embeddings),df[["n_w","n_s","n_c"]]], axis=1)
```

```
X_new
```

	0	1	2	3	4	5	6	7	8	9	...	377	
0	-0.036719	-0.075977	-0.012494	-0.026569	-0.027302	0.059690	0.018057	0.017092	0.009097	-0.017147	...	-0.002832	-0
1	0.000809	-0.011774	-0.024668	-0.013851	0.042000	-0.004458	0.054191	-0.037227	0.062161	-0.022968	...	0.024998	-0
2	0.010068	-0.004346	-0.000853	0.005554	-0.081639	-0.049284	-0.025041	0.039795	0.041797	-0.043278	...	-0.023813	0
3	0.006883	-0.046947	-0.049527	-0.010081	-0.017726	-0.069042	0.014642	0.025617	-0.020526	-0.030805	...	0.073519	-0
4	-0.024798	-0.058147	0.058895	-0.019508	0.016510	-0.078348	-0.142908	-0.036995	-0.023168	-0.022636	...	-0.027766	0
...	...	...	...	...	...	...	...	...	...	...	...	...	
4995	-0.008571	0.079681	-0.018899	-0.001861	-0.077694	-0.096329	0.055912	-0.053277	-0.016167	0.009940	...	-0.021826	0
4996	-0.064108	0.057621	0.095225	-0.035204	0.021385	-0.050942	0.103024	-0.074215	0.074222	0.040593	...	-0.053427	-0
4997	0.039258	0.036492	-0.021289	-0.051159	-0.079778	-0.003736	-0.049690	0.070004	-0.063574	-0.018668	...	0.015649	0
4998	-0.087136	-0.005446	0.020451	0.068807	-0.009414	-0.018395	0.171198	-0.017964	-0.037767	-0.092763	...	-0.015286	-0
4999	0.010728	-0.023654	0.058523	0.026812	-0.065386	-0.091657	0.141122	0.073431	0.008429	0.023026	...	-0.013210	-0

5000 rows × 387 columns

```
X_new.columns=[f"f{i}" for i in range(len(X_new.columns))]
```

```
# min max scaler
from sklearn.preprocessing import MinMaxScaler
scaler=MinMaxScaler()
X=scaler.fit_transform(X_new)
```

```
x_df=pd.DataFrame(X)
```

```
x_df
```

	0	1	2	3	4	5	6	7	8	9	...	377	378
0	0.492873	0.214210	0.485070	0.365148	0.357505	0.731027	0.438363	0.542738	0.502959	0.461707	...	0.488209	0.132130
1	0.596808	0.394395	0.448255	0.409578	0.580055	0.522454	0.518197	0.368965	0.671973	0.440499	...	0.576576	0.162620
2	0.622453	0.415242	0.520274	0.477369	0.183014	0.376707	0.343146	0.615369	0.607112	0.366491	...	0.421590	0.632755
3	0.613632	0.295682	0.373082	0.422748	0.388259	0.312467	0.430820	0.570011	0.408602	0.411942	...	0.730642	0.440928
4	0.525889	0.264249	0.700952	0.389816	0.498201	0.282210	0.082741	0.369708	0.400187	0.441707	...	0.409039	0.547073
...	...	...	...	...	...	...	...	...	...	...	...	...	...
4995	0.570830	0.651066	0.465703	0.451462	0.195682	0.223745	0.521998	0.317616	0.422487	0.560408	...	0.427901	0.631873
4996	0.417018	0.589153	0.810812	0.334985	0.513854	0.371316	0.626084	0.250635	0.710390	0.672105	...	0.327560	0.389714
4997	0.703294	0.529854	0.458474	0.279246	0.188991	0.524803	0.288689	0.712010	0.271489	0.456166	...	0.546890	0.707440
4998	0.353239	0.412154	0.584696	0.698333	0.414951	0.477141	0.776704	0.430590	0.353689	0.186175	...	0.448667	0.230691
4999	0.624281	0.361054	0.699825	0.551631	0.235206	0.238936	0.710254	0.722974	0.500829	0.608095	...	0.455256	0.262249

5000 rows × 387 columns

```
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score, calinski_harabasz_score, davies_bouldin_score
```

KMeans

```
from sklearn.manifold import TSNE

# Apply t-SNE for dimension reduction
tsne = TSNE(n_components=2, random_state=42, perplexity=60,learning_rate=100) # You can adjust n_components and perplexity
X_tsne = tsne.fit_transform(X_new)

print("Shape after t-SNE:", X_tsne.shape)
```

```
Shape after t-SNE: (5000, 2)
```

```
from sklearn.cluster import KMeans
kmeans = KMeans(
    n_clusters=4,
    init="k-means++",
    random_state=42,
    n_init=50
)
clusters = kmeans.fit_predict(X_tsne)


# Add cluster labels to dataframe
df['Kmeans'] = clusters

# Unsupervised evaluation
silhouette = silhouette_score(X_tsne, clusters)
calinski = calinski_harabasz_score(X_tsne, clusters)
davies = davies_bouldin_score(X_tsne, clusters)
```

```
print(f"Silhouette Score: {silhouette:.3f}")
print(f"Calinski-Harabasz Score: {calinski:.3f}")
print(f"Davies-Bouldin Score: {davies:.3f}")

# Inspect cluster examples
for i in range(4):
    print(f"\nCluster {i} examples:")
    print(df[df['Kmeans'] == i]['t'].sample(5).to_list())
```

```
Silhouette Score: 0.488
Calinski-Harabasz Score: 6494.805
Davies-Bouldin Score: 0.666
```

```
Cluster 0 examples:
['do i tick off da loan criteria?', 'last payment—did it go thru or nah', 'This was not authorized by me. Period.', 'how com

Cluster 1 examples:
['What are the minimum credit requirements to apply for a loan?', '$700 TO "VIP EVENTS"? I WATCH NETFLIX IN PAJAMAS—STOP THI

Cluster 2 examples:
['tell me dats a joke', 'hey dude, reset my banking pw', 'how to avoid overdraft?', 'my acc got hacked pls help!!!', 'any mo

Cluster 3 examples:
['there's a $50 charge i don't recognize—pls fix', 'I'm considering a loan—what options do you have?', 'I seek the current i
```

Agglomerative

```
from sklearn.decomposition import TruncatedSVD
from sklearn.cluster import AgglomerativeClustering
```

```
# ==== 4. Agglomerative Clustering ====
n_clusters = 4  # you can change this
agg_clust = AgglomerativeClustering(n_clusters=n_clusters)
cluster_labels = agg_clust.fit_predict(X_tsne)

df['Agg_cluster'] = cluster_labels

# ==== 5. Evaluation ====
sil_score = silhouette_score(X_tsne, cluster_labels)
calinski_score = calinski_harabasz_score(X_tsne, cluster_labels)
davies_score = davies_bouldin_score(X_tsne, cluster_labels)

print(f"Silhouette Score: {sil_score:.3f}")
print(f"Calinski-Harabasz Score: {calinski_score:.3f}")
print(f"Davies-Bouldin Score: {davies_score:.3f}")
```

```
Silhouette Score: 0.453
Calinski-Harabasz Score: 6572.894
Davies-Bouldin Score: 0.710
```

```
from sklearn.manifold import TSNE
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score, calinski_harabasz_score, davies_bouldin_score
import pandas as pd

results = []

# Define parameter grid
perplexities = [5, 10,15,20,25,30,40,50,60,80]
learning_rates = [10,20,30,40,50,70,90,100]
n_clusters_list = [2,3,4,5,6,7]

for perplexity in perplexities:
    for lr in learning_rates:
        # Apply t-SNE
        tsne = TSNE(n_components=2, perplexity=perplexity, learning_rate=lr, random_state=42)
        X_tsne = tsne.fit_transform(X_new)

        for n_clusters in n_clusters_list:
            kmeans = KMeans(n_clusters=n_clusters, random_state=42, n_init=20)
            clusters = kmeans.fit_predict(X_tsne)

            # Evaluate
            silhouette = silhouette_score(X_tsne, clusters)
            calinski = calinski_harabasz_score(X_tsne, clusters)
            davies = davies_bouldin_score(X_tsne, clusters)

            results.append({
                'perplexity': perplexity,
                'learning_rate': lr,
                'n_clusters': n_clusters,
                'silhouette': silhouette,
                'calinski': calinski,
                'davies': davies
            })
```

```
# Convert results to DataFrame
results_df = pd.DataFrame(results)

# Rank by silhouette (higher is better)
best_params = results_df.sort_values(by="silhouette", ascending=False).head(5)
print(best_params)
```

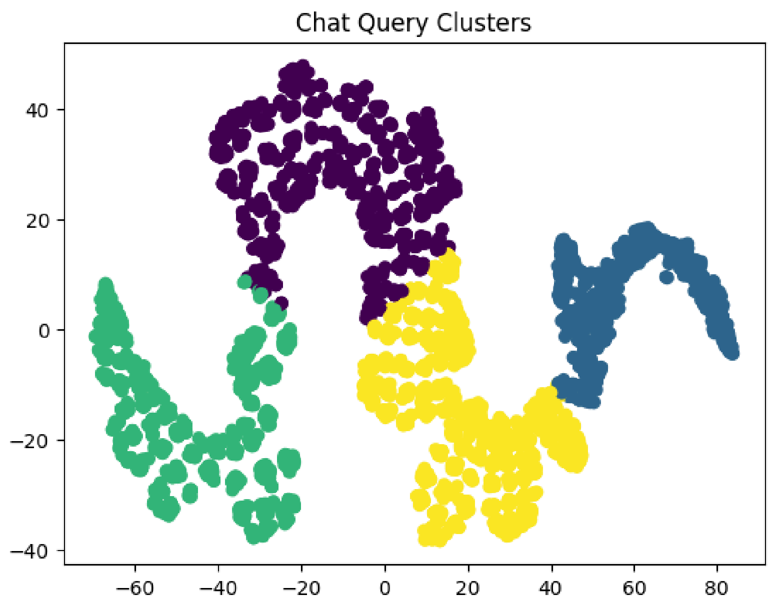
```
     perplexity  learning_rate  n_clusters  silhouette     calinski    davies
428          60            100           4    0.502517  7893.368652  0.679381
416          60             70           4    0.499660  7624.535645  0.690614
470          80             90           4    0.499212  8536.618164  0.676886
464          80             70           4    0.498247  8186.384277  0.681496
476          80            100           4    0.497959  8470.266602  0.682969
```

```
df
```

	t	i	n_c	n_w	n_s	Kmeans	Agg_cluster	HBDSCAN
0	Could you please help me reset my account pass...	password_reset	51	10	1	3	0	2
1	What company charged my account last?	transaction_query	37	7	1	0	2	2
2	How do I schedule an appointment to discuss lo...	loan_inquiry	67	13	1	1	3	3
3	Which loans 2day suited me?	loan_inquiry	27	6	1	2	1	-1
4	Do you have any updated timelines for my loan ...	loan_inquiry	55	11	1	3	0	2
...	...	...	...	...	...	...	...	...
4995	Was the transaction processed correctly?	transaction_query	40	6	1	0	0	2
4996	Apple Pay charges I didn't make—fraud!!	fraud_report	39	10	2	0	2	2
4997	Can you provide detailed steps on how to begin...	loan_inquiry	66	13	1	1	3	3
4998	what's the acc say? nervous chuckle	balance_inquiry	35	9	2	0	2	2
4999	why'd my transac get dunked on?	transaction_query	31	9	1	0	2	-1

5000 rows × 8 columns

```
from sklearn.manifold import TSNE
import matplotlib.pyplot as plt

tsne = TSNE(n_components=2, random_state=42, perplexity=60,learning_rate=100) # You can adjust n_components and perplexity
vis_data = tsne.fit_transform(X_new)

plt.scatter(vis_data[:, 0], vis_data[:, 1], c=df['Kmeans'], cmap='viridis')
plt.title("Chat Query Clusters")
plt.show()
```

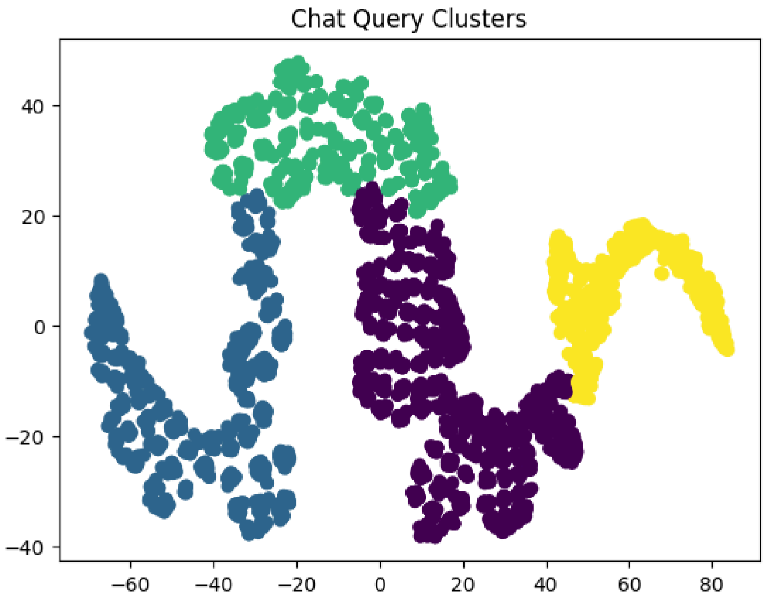



```
# Inspect cluster examples
for i in range(4):
```

```
    print(f"\nCluster {i} examples:")
    print(df[df['Kmeans'] == i]['t'].sample(5).to_list())
```

```
Cluster 0 examples:
['How do I call up the loan app process?', 'Was the last payment processed correctly?', 'i need ta check how much mony i hav

Cluster 1 examples:
['Would you confirm my eligibility for a turnkey project loan?', 'Can you please confirm the current phase of my loan applic

Cluster 2 examples:
['new cust loan apply? how?', 'cant login, pass sucks now', 'How do I go abt gettin a loan?', 'transac went poof—why?', 'no

Cluster 3 examples:
['Was the previous transaction successful without issues?', 'Can you tell me the time of my last transaction?', 'Did the mos
```

```
from sklearn.manifold import TSNE
import matplotlib.pyplot as plt

#tsne = TSNE(n_components=2, random_state=42, perplexity=60,learning_rate=100) # You can adjust n_components and perplexity
#vis_data = tsne.fit_transform(X_new)

plt.scatter(vis_data[:, 0], vis_data[:, 1], c=df['Agg_cluster'], cmap='viridis')
plt.title("Chat Query Clusters")
plt.show()
```




```
# Inspect cluster examples
for i in range(4):
    print(f"\nCluster {i} examples:")
    print(df[df['Agg_cluster'] == i]['t'].sample(5).to_list())
```

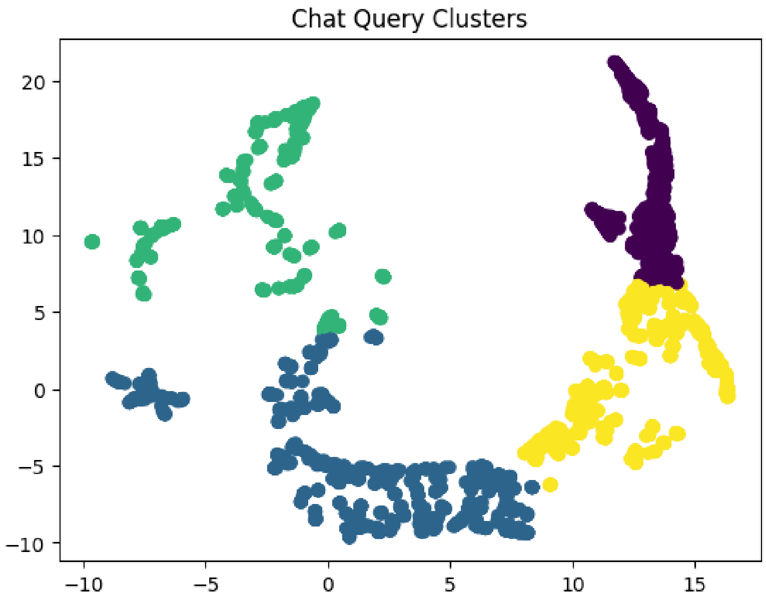
```
Cluster 0 examples:
['This is unacceptable—ANSWER MY SAVINGS INQUIRY!', 'What are the requirements to apply for an unsecured loan?', 'What's the

Cluster 1 examples:
['do i meet payday alt criteria?', 'do i check ur loan boxes?', 'do i ring as loan material?', 'still no reset email, resenc

Cluster 2 examples:
['recent transction? cries silently', 'someone's takin loans in my name—HELP!!', 'Loan application—how do I get started?',

Cluster 3 examples:
['$450 AT "FITNESS FREAK"? I HATE GYMS—REVERSE THIS OR I'M CLOSING MY ACCOUNT!', 'WHO'S "LUXURY LIVING"? I LIVE IN A STUDIO-
```

```
# umap dimensionality reduction
import umap
import pandas as pd
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score, calinski_harabasz_score, davies_bouldin_score

# Step 1: Apply UMAP
umap_model = umap.UMAP(
    n_components=2,       # project to 2D
```

```
    n_neighbors=15,       # size of local neighborhood (controls balance of local/global structure)
    min_dist=0.1,         # how tightly UMAP packs points together
    random_state=42
)
X_umap = umap_model.fit_transform(X_new)

print("Shape after UMAP:", X_umap.shape)

# Step 2: Cluster with KMeans
kmeans = KMeans(n_clusters=4, random_state=42, n_init=20)
clusters = kmeans.fit_predict(X_umap)

# Step 3: Add cluster labels to dataframe
df["UMAP_KMeans"] = clusters

# Step 4: Evaluate clustering
silhouette = silhouette_score(X_umap, clusters)
calinski = calinski_harabasz_score(X_umap, clusters)
davies = davies_bouldin_score(X_umap, clusters)

print(f"Silhouette Score: {silhouette:.3f}")
print(f"Calinski-Harabasz Score: {calinski:.3f}")
print(f"Davies-Bouldin Score: {davies:.3f}")
```

```
Shape after UMAP: (5000, 2)
Silhouette Score: 0.488
Calinski-Harabasz Score: 6494.805
Davies-Bouldin Score: 0.666
```

```
plt.scatter(X_umap[:, 0], X_umap[:, 1], c=df['UMAP_KMeans'], cmap='viridis')
plt.title("Chat Query Clusters")
plt.show()
```




```
# Inspect cluster examples
for i in range(4):
    print(f"\nCluster {i} examples:")
    print(df[df['UMAP_KMeans'] == i]['t'].sample(5).to_list())
```

```
Cluster 0 examples:
['What are the steps to apply for a startup seed funding loan?', 'How do I request a credit card application package by mail

Cluster 1 examples:
['last transction—am i in debt now', 'Did that last payment go through or not?', 'do i got shiine 4 loan approval?', 'check

Cluster 2 examples:
['card said no. why?', 'did i pass prequal or nah?', 'this not how i roll', 'atm hacked or smth', 'gotta block dis ASAP']

Cluster 3 examples:
['Please advise on how I may reset my password.', 'What was the time and date on my last purchase?', 'That check deposit was
```