PartA

March 14, 2025

Gracie Longman F434523

1 Part A

[1]:
```
from google.colab import drive
drive.mount("/content/drive")
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call
drive.mount("/content/drive", force_remount=True).
```

2 Question 1

[2]:
```
# Downloads and imports for question 1
import string
import pandas as pd
import nltk
from textblob import TextBlob
import re
from nltk.tokenize import word_tokenize
from nltk.stem.porter import PorterStemmer
from nltk.corpus import stopwords
nltk.download('popular')
nltk.download('punkt')
nltk.download('punkt_tab')
```

```
[nltk_data] Downloading collection 'popular'
[nltk_data]    |
[nltk_data]    | Downloading package cmudict to /root/nltk_data…
[nltk_data]    |   Package cmudict is already up-to-date!
[nltk_data]    | Downloading package gazetteers to /root/nltk_data…
[nltk_data]    |   Package gazetteers is already up-to-date!
[nltk_data]    | Downloading package genesis to /root/nltk_data…
[nltk_data]    |   Package genesis is already up-to-date!
[nltk_data]    | Downloading package gutenberg to /root/nltk_data…
[nltk_data]    |   Package gutenberg is already up-to-date!
[nltk_data]    | Downloading package inaugural to /root/nltk_data…
[nltk_data]    |   Package inaugural is already up-to-date!
```

```
[nltk_data]    | Downloading package movie_reviews to
[nltk_data]    |     /root/nltk_data…
[nltk_data]    |   Package movie_reviews is already up-to-date!
[nltk_data]    | Downloading package names to /root/nltk_data…
[nltk_data]    |   Package names is already up-to-date!
[nltk_data]    | Downloading package shakespeare to /root/nltk_data…
[nltk_data]    |   Package shakespeare is already up-to-date!
[nltk_data]    | Downloading package stopwords to /root/nltk_data…
[nltk_data]    |   Package stopwords is already up-to-date!
[nltk_data]    | Downloading package treebank to /root/nltk_data…
[nltk_data]    |   Package treebank is already up-to-date!
[nltk_data]    | Downloading package twitter_samples to
[nltk_data]    |     /root/nltk_data…
[nltk_data]    |   Package twitter_samples is already up-to-date!
[nltk_data]    | Downloading package omw to /root/nltk_data…
[nltk_data]    |   Package omw is already up-to-date!
[nltk_data]    | Downloading package omw-1.4 to /root/nltk_data…
[nltk_data]    |   Package omw-1.4 is already up-to-date!
[nltk_data]    | Downloading package wordnet to /root/nltk_data…
[nltk_data]    |   Package wordnet is already up-to-date!
[nltk_data]    | Downloading package wordnet2021 to /root/nltk_data…
[nltk_data]    |   Package wordnet2021 is already up-to-date!
[nltk_data]    | Downloading package wordnet31 to /root/nltk_data…
[nltk_data]    |   Package wordnet31 is already up-to-date!
[nltk_data]    | Downloading package wordnet_ic to /root/nltk_data…
[nltk_data]    |   Package wordnet_ic is already up-to-date!
[nltk_data]    | Downloading package words to /root/nltk_data…
[nltk_data]    |   Package words is already up-to-date!
[nltk_data]    | Downloading package maxent_ne_chunker to
[nltk_data]    |     /root/nltk_data…
[nltk_data]    |   Package maxent_ne_chunker is already up-to-date!
[nltk_data]    | Downloading package punkt to /root/nltk_data…
[nltk_data]    |   Package punkt is already up-to-date!
[nltk_data]    | Downloading package snowball_data to
[nltk_data]    |     /root/nltk_data…
[nltk_data]    |   Package snowball_data is already up-to-date!
[nltk_data]    | Downloading package averaged_perceptron_tagger to
[nltk_data]    |     /root/nltk_data…
[nltk_data]    |   Package averaged_perceptron_tagger is already up-
[nltk_data]    |       to-date!
[nltk_data]    |
[nltk_data]  Done downloading collection popular
[nltk_data] Downloading package punkt to /root/nltk_data…
[nltk_data]   Package punkt 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!
```

[2]: True

[3]:
```
# load_doc, clean_text functions were adapted from https://github.com/gcosma/
↪COP509/blob/main/Tutorials/Tutorial1NLPcleanText.ipynb
# remove_dups and get_ids_ratings was my own work

def load_doc(filename, skip_header=False):
  '''Read the file'''
  # Check file extension
  if filename.endswith('.xlsx'):
      df = pd.read_excel(data_path+filename, header=None)
      df.dropna(inplace=True)
      lines = df.values.tolist()
  else:
      with open(data_path + filename, 'r') as file:
        if skip_header:
          next(file)
        lines = []
        for line in file:
            clean_line = line.strip()
            lines.append(clean_line)
  return lines

def remove_dups(text):
  '''Remove duplicates'''
  reviews=[]
  seen=[]
  # Loop through each review
  for review in text:
    # Check if line starts with numeric (i.e., an ID) followed by alphabetic
    match = re.match(r'^\d+(.+)$', review)
    if match:
      # Remove numeric (i.e. the id)
      rev_content = match.group(1).strip()
    else:
      # Can just use the whole line
      rev_content = review.strip()
    # Compare if the line has been seen or not
    if rev_content not in seen:
      seen.append(rev_content)
      reviews.append(review)
  return reviews

def clean_doc(text):
  '''Clean the text'''
  # split into words
  tokens = word_tokenize(text)
  # correct spelling mistakes
  corrected = [str(TextBlob(word).correct()) for word in tokens]
  # # stem words
  porter = PorterStemmer()
  stemmed = [porter.stem(word) for word in corrected]
  # convert to lower case
  tokens = [w.lower() for w in corrected]
  # remove punctuation from each word
  table = str.maketrans('', '', string.punctuation)
  stripped = [w.translate(table) for w in tokens]
  # remove remaining tokens that are not alphabetic or digit
  words = [word for word in stripped if word.isalnum()]
  # Remove short words but keep numbers
  words = [word for word in words if len(word) > 1 or word.isdigit()]
  # filter out stop words
  stop_words = stopwords.words('english')
  stop_words.remove('not') # I have decided to allow not
  words = [w for w in words if not w in set(stop_words)]
  return words

def get_ids_ratings_text(clean_text):
  '''Get ids, ratings and text'''
  clean_text = [clean_doc(doc) for doc in text]
  ids=[]
  ratings=[]
  for words in clean_text:
    #Store the ids and ratings
    ids.append(words[0])
    #only append numerical part
    rating = ''.join([char for char in words[-1] if char.isdigit()])  # Extract␣
 ↪the number
    ratings.append(rating)

    #remove ids and ratings from the docs as this will not help analysis
    words.pop(0)
    if words:
      if isinstance(words[-1], str):
          # Check if last element contains alphabetic characters
          alpha_chars = ''.join([char for char in words[-1] if char.isalpha()])
          if alpha_chars:
              words[-1] = alpha_chars
          else:
              words.pop(-1)
  return ids, ratings, clean_text

# Label data path
data_path = "/content/drive/My Drive/Colab Notebooks/COP509cw/Datasets/"
```

```
# Load and clean data
text = load_doc("JewelleryReviewsLSA.csv", skip_header=True)
text = remove_dups(text)
ids, ratings, clean_text = get_ids_ratings_text(text)

print(ids)
print(ratings)
print(clean_text)
```

```
['32767', '15959', '43515', '30720', '1816', '265', '54548', '28250', '30773',
'11856', '2185', '36727', '11087', '535', '32496', '44534', '48216', '50609',
'50640', '50650', '27858', '21185', '22946', '9050', '44591', '24452', '12483',
'56342', '33009', '37896', '51030', '13559', '52663', '39606', '2520', '33813',
'42026', '51396', '45548', '45856', '33746', '3865', '42077', '53750', '39496',
'33858', '17442', '47910', '58595', '23979', '39620', '3978', '34483', '48781',
'6421', '53693', '28474', '49855', '10209', '49720', '25080', '43839', '40749',
'6522', '34529', '19647', '45289', '33620', '14499', '1185', '56679', '50467',
'27793', '34266', '48772', '30926', '42604', '11269', '9441', '18988', '19548',
'25378', '10535', '57009', '45203', '6135', '44135', '20090', '8110', '32674',
'4375', '41889', '51907', '38305', '22058', '10612', '17166', '36164', '58481',
'26246', '2033', '48779', '34523', '9726', '56494', '49525', '45278', '35694',
'41876', '17309', '11135', '17273', '11247', '57123', '25299', '55017', '7432',
'2114', '40871', '33251', '17304', '50019', '27679', '6158', '22408', '29722',
'36677', '2780', '17944', '19944', '31657', '52867', '49216', '40373', '28648',
'37486', '30640', '2131', '19852', '2134', '36585', '26535', '51474', '21070',
'56330', '53660', '44126', '13373', '17607', '41459', '54748', '33571', '45860',
'46500', '27474', '43945', '52837', '12358', '41319', '39932', '45146', '50197',
'8341', '52375', '209', '28542', '216', '47345', '11356', '33632', '38637',
'7110', '6649', '51356', '44358', '36165', '943', '37864', '642', '10642',
'37794', '45518', '3494', '735', '10037', '41872', '28543', '53409', '56865',
'44489', '44490', '56830', '3']
['1', '1', '1', '1', '1', '1', '1', '2', '2', '2', '2', '2', '2', '2', '3', '3',
'3', '3', '3', '3', '3', '3', '3', '3', '3', '3', '3', '4', '4', '4', '4', '4',
'4', '4', '4', '4', '4', '4', '4', '4', '4', '4', '4', '4', '5', '5', '5', '5',
'5', '5', '5', '5', '5', '5', '5', '5', '5', '5', '5', '5', '5', '5', '5', '5',
'5', '5', '5', '5', '5', '5', '5', '5', '5', '5', '5', '5', '5', '5', '5', '5',
'5', '5', '5', '5', '5', '5', '5', '5', '5', '5', '5', '5', '5', '5', '5', '5',
'5', '5', '5', '4', '5', '5', '5', '5', '5', '5', '5', '5', '5', '4', '5', '5',
'5', '1', '2', '1', '1', '2', '1', '5', '5', '5', '2', '5', '3', '4', '5', '5',
'5', '5', '5', '4', '4', '5', '5', '4', '5', '5', '5', '5', '304', '4', '5',
'5', '5', '5', '5', '1', '1', '1', '2', '1', '3', '4', '4', '5', '5', '5', '5',
'5', '5', '5', '5', '4', '1', '2', '1', '1', '1', '2', '3', '4', '33', '1', '2',
'3', '1', '2', '5', '3', '4', '2', '5', '5', '1', '5', '2', '3', '5', '5', '5',
'', '3']
[['expect', 'like', 'regular', 'size', 'ring', 'one', 'look', 'like', 'ring',
'toy', 'something', 'funny', 'rings', '5mm', 'ring', 'may', '1mmso',
'ridiculousmartin15', 'itto', 'round', 'diamond', 'voltaire', 'king', 'ask',
'white', 'old'], ['ordered', 'ring', 'stated', 'toe', 'king', 'description',
'ring', 'came', 'quickly', 'not', 'toe', 'ring', 'nt', 'know', 'anyone', 'size',
'8', 'toe', 'know', 'anyone', 'please', 'direct', 'seller', 'wedding', 'band',
'toe', 'ring', 'thumb', 'ring', 'sick', 'one'], ['ring', 'beautiful', 'ring',
'first', 'shipment', 'ring', 'scratches', 'diamond', 'missing', '2', 'diamonds',
'shoulder', 'ring', 'returned', 'ring', 'replacement', 'ring', 'second', 'ring',
'even', 'worse', 'clearly', 'visible', 'white', 'scratch', 'right', 'middle',
'black', 'diamond', 'not', 'look', 'great', 'time', '1', 'diamond', 'missing',
'shoulder', 'ring', 'returned', 'refund', 'thanks', 'clot', 'amazon'], ['huge',
'waste', 'money', 'month', 'part', 'holds', 'symbol', 'necklace', 'broke',
'seriously', 'not', 'wasting', 'money', 'low', 'quality', 'item'], ['quality',
'look', 'not', 'anticipated', 'fliesi', 'would', 'not', 'recommend', 'item'],
['quality', 'item', 'not', 'expectationsthe', 'top', 'scratched', 'hinges',
'not', 'line', 'predrilled', 'holes', 'staining', 'inconsistent', 'saw', 'item',
'store', 'would', 'not', 'purchased'], ['hard', 'ca', 'nt', 'wear', 'material',
'hard', 'not', 'easy', 'wear', 'nt', 'like', 'saved', 'money', 'got'],
['wanted', 'know', 'ring', 'like', '2', 'rings', 'one', 'ring', 'beyond',
'gorgeous', 'love'], ['vice', 'ring', 'expensive', 'ring', 'one', 'stone',
'missing', 'not', 'worth', 'returning', 'would', 'pay', 'postage', 'ring',
'cost'], ['wore', 'toe', 'ring', 'one', 'day', 'poof', 'stone', 'gone', 'sits',
'pile', 'broken', 'sterling', 'silver', 'rings', 'things'], ['king', 'way',
'small', 'looks', 'like', 'toy', 'putting', 'would', 'not', 'recommend', 'want',
'nice', '12', 'cart', 'ring'], ['week', 'one', 'jewels', 'fell', 'wing',
'metal', 'already', 'vanishing', 'spend', 'money', 'higher', 'quality', 'item'],
['wanted', 'not', 'favorite', 'piercing', 'mine', 'wear', 'bioplast', 'cut',
'break', 'certain', 'metals'], ['serves', 'purpose', 'seemed', 'image', 'lot',
'prettier', 'sparkling', 'turned', 'wear', 'shirt', 'since', 'not',
'compliment', 'anything', 'wear'], ['ring', 'beautiful', 'ring', 'first',
'shipment', 'ring', 'scratches', 'diamond', 'missing', '2', 'diamonds',
'shoulder', 'ring', 'returned', 'ring', 'replacement', 'ring', 'second', 'ring',
'even', 'worse', 'clearly', 'visible', 'white', 'scratch', 'right', 'middle',
'black', 'diamond', 'not', 'look', 'great', 'time', '1', 'diamond', 'missing',
'shoulder', 'ring', 'returned', 'refund', 'thanks', 'clot', 'amazon'], ['nice',
'small', 'sized', 'ring', 'stick', 'rings', 'different', 'looks'], ['got',
'ring', 'birthday', 'love', 'not', 'imagine', 'woman', 'not', 'adoring',
'ring'], ['first', 'skeptically', 'ring', 'price', 'low', 'nt', 'wear', 'lot',
'gold', 'purchased', 'ring', 'nice', 'gold', 'necklaces', 'wear', 'wanted',
'ring', 'work', 'ring', 'nt', 'look', 'like', '35', 'ring'], ['ring', 'awesome',
'wear', 'ring', 'index', 'finger', 'fits', 'great', 'comfortable', 'ring'],
['not', 'ring', 'beautiful', 'jeweller', 'accommadating', 'ring', 'reach', 'us',
'time', 'appreciate', 'care', 'quick', 'receiving', 'ring', 'thank'], ['love',
'ring', 'not', 'ring', 'would', 'want', 'wear', 'everyday', 'ring', 'claws',
'hold', 'stone', 'sharp', 'get', 'stuck', 'things', 'scratched', 'me'],
['bought', 'ring', 'valentine', 'gift', 'could', 'nt', 'give', 'since', 'hands',
'holding', 'heart', 'symbol', 'would', 'reverse', 'whenever', 'ring', 'spine',
'4', 'ring', 'alternating', 'facing', 'direction', 'cut', 'ring', 'kept'],
['ring', 'little', 'small', 'ring', 'finger', 'would', 'better', 'pink',
'ring'], ['ring', 'pretty', 'enough', 'metal', 'ring', 'substantial', 'pushes',
'easily'], ['husband', 'loves', 'thing', 'ring', 'resided', 'due', 'way',
'ring', 'made'], ['ring', 'pretty', 'flexible', 'turned', 'good', 'ordered',
'ring', 'finger', 'size', 'made', 'look', 'better', 'pink', 'ring'],
['definitely', 'help', 'lessen', 'appetite', 'however', 'ears', 'sore',
'wearing', '3', 'hours', 'next', 'days', 'tried', 'wear', 'increase', 'wearing',
'time', 'good', 'pain', 'tolerable', 'may', 'not', 'notice', 'discomfort',
'ears', 'loves', 'swollen', 'stop', 'wearing', '4', 'days'], ['got', '3',
'belly', 'rings', '3', 'tongue', 'rings', '3', 'lip', 'rings', 'fig', 'car',
'stuff', 'got', 'cool', 'love', 'pink', 'peace', 'sign', 'tongue', 'ring',
'complaint', 'got', '2', 'yellow', 'lip', 'rings', 'not', 'much', 'yellow',
'overall', 'happy'], ['similar', 'ring', 'auction', 'site', 'flaw', 'lies',
'middle', 'part', 'ring', 'complete', 'ring', 'cuts', 'flexibility', 'joint',
'ended', 'taking', 'mine', 'silver', 'smith', 'remove', 'back', 'section',
'middle', 'ring', 'really', 'helped', 'flexibility'], ['wish', 'ring', 'wish',
'cut', 'portion', 'went', 'way', 'around', 'ring', 'great', 'comfortable',
'ring'], ['ring', 'shipped', 'accordingly', 'love', 'shipping', 'pace', 'ring',
'beautiful', 'thing', 'nt', 'care', 'line', 'looks', 'like', 'ring', 'may',
'resized', 'including', 'specification', 'included', 'ring', 'met',
'expectations', 'especially', 'beginning', 'ring'], ['husband', 'wears', 'size',
'10', 'ring', 'tight', 'new', 'thumb', 'ring', 'guess', 'nt', 'order', 'things',
'online', 'cut', 'ring'], ['told', 'ring', 'comfortable', 'wear', 'quite',
'surprised', 'please', 'see', 'masonic', 'ring', 'titanic'], ['nt', 'even',
'ordered', 'ring', 'read', 'reviews', 'notice', 'picture', '1', 'pictures',
'23', 'two', 'different', 'rings', 'going', 'order', 'ring', 'seems', 'ring',
'received', 'andrd', 'ring', 'smaller', 'cross', 'design'], ['picture', 'rings',
'shows', 'blue', 'sapphires', 'disappointed', 'see', 'ring', 'received', 'dark',
'stones', 'appear', 'black', 'ring', 'definitely', 'not', 'advertised'],
['favorite', 'ring', 'also', 'one', 'first', 'rings', 'ever', 'bought', 'new',
'buying', 'peter', 'rings', 'something', 'may', 'help', 'letter', 'rings',
'tend', 'bit', 'bulkier', 'types', 'rings', 'metal', 'softer', 'last', 'liked',
'ring', 'design', 'dead', 'bags', 'snakes', 'catches', 'lot', 'attention'],
['product', 'came', 'fast', 'like', 'amazon', 'explained', 'ring', 'clearly',
'written', 'war', 'thick', 'likebut', 'guess', 'like', 'small', 'rings',
'ring'], ['nice', 'small', 'sized', 'ring', 'stick', 'rings', 'different',
'looks'], ['attractive', 'high', 'quality', 'item', 'young', 'teenager',
'small', 'adult'], ['could', 'see', 'quality', 'work', 'rosary', 'pleased',
'opened', 'container', 'could', 'immediately', 'smell', 'olive', 'wood', 'one',
'thing', 'buyers', 'need', 'aware', 'buying', 'item', 'nt', 'mentioned',
'description', 'holy', 'present', 'compartment', 'rosary', 'opinion', 'adds',
'quality', 'rosary', 'not', 'sure', 'seller', 'nt', 'describe', 'fact',
'looking', 'items', 'seller'], ['love', 'ring', 'complaint', 'metal', 'soft',
'bent', 'point', 'wear', 'anymore', 'still', 'wear', 'necklace', 'love',
'occasion', 'bluish', 'onto', 'figure', 'took', 'less', 'two', 'months',
'almost', 'impossible', 'wear'], ['sparkle', 'pretty', 'dainty', 'must',
'looking', 'for'], ['solidbeautiful', 'ring', 'expecting', 'color', 'picture',
'disappointed', 'barely', 'pink', 'first', 'saw', 'thought', 'lavender',
'still', 'pretty', 'buy', 'design', 'not', 'color'], ['beautiful', 'piece',
'looked', 'long', 'time', 'replace', 'hand', 'given', 'egypt', 'one', 'perfect',
'delicate', 'sturdy', 'lovely', 'look', 'detail', 'pretty', 'elegant', 'wear',
'often'], ['bought', 'two', 'rings', 'fit', 'either', 'side', 'platino',
'princess', 'wedding', 'ring', 'help', 'anchor', 'add', 'ring', 'perfect',
'fit', 'lot', 'compliments', 'new', 'ring'], ['ring', 'perfect', 'say', 'spend',
'thousands', 'nt', 'ring', 'shines', 'perfectly', 'love', 'ring'], ['addition',
'wonderful', 'sending', 'ring', 'ring', 'beautiful', 'daughter', 'thrilled',
'ring', 'thank', 'dorothy'], ['girlfriend', 'especially', 'enjoys', 'ring',
'thickness', 'ring', 'band', 'past', 'similar', 'rings', 'always', 'frustrated',
'flies', 'ring', 'band', 'ring', 'nice', 'solid', 'band'], ['absolutely',
'love', 'ring', 'got', 'engagement', 'ring', 'web', '09', 'ring', 'beautiful',
'unable'], ['love', 'ring', 'flowers', 'go', 'way', 'around', 'ring', 'fits',
'ring', 'finger', 'ever', 'find', 'right', 'nt', 'buy', 'engagement', 'ring',
'use', 'engagement', 'ring', 'not', 'getting', 'debt', 'marriage'], ['ring',
'gorgeous', 'fits', 'right', 'sterling', 'silver', 'celtic', 'rings', 'little',
'worried', 'first', 'size', 'band', 'since', '10', 'wear', '9', 'band', 'wide',
'ring', 'fits', 'perfectly', 'hardly', 'loose', 'even', 'though', 'man', 'ring',
'nt', 'big', 'clung', 'must', 'lovers', 'celtic', 'rings', 'ring', 'also',
'arrived', 'quickly', 'ring', 'bought', 'company', 'would', 'definitely',
'business'], ['ought', 'ring', 'thumb', 'ring', 'everyone', 'notices',
'comments', 'pretty', 'cool', 'looks', 'spin', 'want', 'know', 'got', 'would',
'recommend', 'ring', 'one', 'also', 'come'], ['love', 'ring', 'shines',
'beautiful', 'looks', 'real', 'anyone', 'like', 'buy', 'one', 'ring', 'stones',
'good', 'size', 'friends', 'love', 'ring', 'could', 'pass', 'real', 'diamond',
'love', 'ring', 'worth', 'money', 'better', 'ring', 'bought', 'von', 'silver',
'cubic', 'shine', 'ring', 'ring', 'bought', '3', 'stones', 'worth', 'price',
'anne', 'marie'], ['owned', 'triple', 'roll', 'ring', 'three', 'rings',
'slowly', 'broke', 'purchased', 'far', 'none', 'individual', 'rings', 'broken',
'yet'], ['love', 'rings', 'need', 'pink', 'rings', 'hard', 'find', 'size', '4',
'rings', 'nt', 'children', 'ring', 'nice', 'nice', 'flower', 'ring', 'perfect',
'pink', 'ring', 'me'], ['bought', 'ring', 'old', 'date', 'beautiful', 'ring',
'fair', 'price', 'guy', 'looking', 'save', 'money', 'afford', 'engagement',
'ring', 'definitely', 'get', 'this'], ['ring', 'lovely', 'advertised', 'pink',
'ring', 'actually', 'wear', 'thumb', 'ring'], ['unfortunately', 'small',
'woman', 'ring', 'size', '4', 'almost', 'impossible', 'purchase', 'rings',
'without', 'sized', 'purchased', 'ring', 'would', 'something', 'wear',
'vacation', 'things', 'could', 'otherwise', 'harm', 'expensive', 'rings',
'dont', 'even', 'need', 'expensive', 'rings', 'ring', 'fits', 'looks',
'beautiful'], ['works', 'perfectly', 'rings', 'wider', 'rings', 'narrow',
'wide'], ['ring', 'absolutely', 'stunning', 'beautiful', 'would', 'recommend',
'amethyst', 'ring', 'anyone', 'market', 'reasonably', 'prices', 'amethyst',
'ring'], ['recently', 'lost', 'ring', 'much', 'like', 'one', 'searching',
'high', 'low', 'another', 'ring', 'replace', 'one', 'lost', 'came', 'across',
'ring', 'ring', 'absolutely', 'brilliant', 'ring', 'look', 'shines', 'sparkles',
'like', 'little', 'diamonds', 'love', 'ring', 'would', 'recommend', 'anyone',
'wants', 'ring', 'element', 'adorable', 'time'], ['still', 'wearing', 'pink',
'ring', 'trinity', 'knotfinally', 'pink', 'ring', 'not', 'turn', 'finger',
'green'], ['got', 'not', 'engagement', 'ring', 'much', 'surprise', 'two',
'rings', 'fit', 'together', 'make', '1', 'looked', 'like', 'one', 'ring',
'picture', 'must', 'say', 'beautiful', 'ring', 'always', 'catches', 'peoples',
'attention'], ['love', 'ring', 'suggest', 'every', 'girl', 'ring', 'jewel',
'collection'], ['wife', 'needed', 'ring', 'wear', 'since', 'actual', 'wedding',
'ring', 'lost', 'not', 'expecting', 'one', 'million', 'dollar', 'ring',
'course', 'arrived', 'not', 'turned', 'heads', 'coworkers', 'actually',
'noticed', 'ring', 'ever', 'noticed', 'old', 'ring', 'cubic', 'zirconia',
'looks', 'authentic', 'enough', 'basic', 'scrutiny', 'may', 'never', 'go',
'back', 'diamond', 'ring', 'wedding', 'ring', 'always', 'silver', 'band',
'believe', 'ring', 'truly', 'defines', 'subliminal', 'misconception',
'deserves', 'diamondgreat', 'temporary', 'ring', 'care', 'price', 'perfect',
'choice', 'want', 'turn', 'heads', 'without', 'breaking', 'bank'], ['checked',
'around', 'jewel', 'stores', 'prices', 'could', 'nt', 'find', 'rings', 'could',
'even', 'compare', 'not', 'huge', 'ring', 'definitely', 'not', 'microscopic',
'either', 'bought', 'ring', 'go', 'along', 'solitary', 'engagement', 'ring',
'makes', 'perfect', 'replacement', 'plain', 'wedding', 'band', 'really', 'glad',
'decided', 'go', 'ring', 'even', 'though', 'kind', 'nervous', 'ordering',
'ring', 'online'], ['bought', '17', 'year', 'old', 'daughter', 'seen', 'small',
'heart', 'shaped', 'ring', 'coach', 'store', 'liked', 'style', 'one', 'much',
'cheaper', 'ended', 'much', 'prettier', 'really', 'eye', 'catching', 'ring',
'nt', 'look', 'cheap', 'sparkles', 'beautifully', 'actually', 'ordering',
'second', 'ring', 'since', 'leaving', 'college', 'next', 'month', 'worried',
'losing', 'think', 'like', 'knowing', 'spare', 'ring', 'home', 'case', 'price',
'not', 'big', 'deal', 'wears', 'ring', 'right', 'ring', 'finger', 'instead',
'high', 'school', 'ring', 'loves', 'much', 'would', 'suggest', 'ring', 'fan',
'hearts', 'little', 'bit'], ['love', 'ring', 'wear', 'time', 'bought', 'wear',
'husband', 'go', 'vacation', 'nt', 'worry', 'losing', 'real', 'ring', 'stone',
'already', 'lose', '50', 'ring', 'not', '5000', 'ring', 'almost', 'every',
'time', 'leave', 'house', 'get', 'compliment', 'ring', 'mon', 'not', 'tell',
'different', 'ring', 'real', 'wedding', 'ring', 'little', 'bulkier', 'picture',
'leads', 'one', 'believe', 'still', 'wonderful', 'ring', 'anything', 'happens',
'stone', 'fall', 'probably', 'buy', 'another', 'ring', 'much', 'love', 'ring'],
['ring', 'exactly', 'wanted', 'actually', 'bought', 'another', 'nt', 'quite',
'solid', 'made', 'ring', 'thanks', 'great', 'ring'], ['gorgeous', 'ring',
'holder', 'sits', 'dresser', 'holds', 'wedding', 'ring', 'every', 'night',
'gorgeous', 'love'], ['ordered', 'ring', 'last', 'december', 'opened', 'ring',
'box', 'arrived', 'absolutely', 'took', 'breath', 'away', 'gorgeous', 'ring',
'pictures', 'nt', 'justice', 'size', 'color', 'stone', 'suit', 'taste',
'perfectly', 'quality', 'ring', 'excellent', 'quite', 'variety', 'silver',
'rings', 'ring', 'one', 'wear', 'regular', 'basis', 'taste', 'rings', 'toward',
'boldunique', 'side', 'ring', 'definitely', 'fits', 'bill'], ['bought', 'ring',
'size', '3', 'toe', 'ring', 'came', 'much', 'sooner', 'expected', 'first',
'surprise', 'second', 'surprise', 'absolutely', 'cut', 'heartshaped', 'box',
'bow', 'top', 'ring', 'came', 'third', 'surprise', 'ring', 'fit', 'perfectly',
'totally', 'true', 'picture', 'looked', 'even', 'better', 'person', 'already',
'shopping', 'second', 'dem', 'avenue', 'ring', 'buds', 'dem', 'avenue',
'supplying', 'small', 'rings', 'people', 'like', 'fitted', 'toe', 'rings'],
['unfortunately', 'small', 'woman', 'almost', 'impossible', 'purchase', 'rings',
'without', 'sized', 'purchased', 'ring', 'would', 'something', 'wear',
'vacation', 'things', 'could', 'otherwise', 'harm', 'expensive', 'rings',
'dont', 'even', 'need', 'expensive', 'rings', 'ring', 'stunning', 'delicate',
'beautiful', 'ring', 'fits', 'looks', 'awesum'], ['ring', 'right', 'one',
'everyone', 'want', 'beautiful', 'ring', 'hand', 'catch', 'everyone',
'attention'], ['great', 'ring', 'tried', 'different', 'one', 'first',
'individual', 'rings', 'thin', 'snapped', 'within', 'weeks', 'rings', 'one',
'thicker', 'making', 'bolder', 'unable', 'ring', 'course', 'costs', 'bit', 'ca',
'nt', 'go', 'wrong', 'paying', 'better', 'quality'], ['neck', 'ring', 'timely',
'manner', 'looks', 'antique', 'would', 'recommend', 'ring', 'garden', 'lover'],
['fussy', 'attractive', 'ring', 'purchased', 'replacement', 'lost', 'wedding',
'ring', 'stacks', 'beautifully', 'engagement', 'ring', 'also', 'perfect',
'catching', 'light', 'appealing', 'luminosity'], ['second', 'engagement',
'king', 'got', 'birthday', 'love', 'nice', 'shiny', 'must', 'buy', 'ring',
'anyone'], ['love', 'ring', 'snug', 'thickness', 'ring', 'wanting', 'buy',
'ring', 'buy', 'size', 'largerthe', 'ring', 'beautiful', 'although', 'bought',
'bc', 'boys', 'autismi', 'using', 'new', 'wedding', 'band', 'since', 'go',
'rings', 'ofteni', 'nt', 'think', 'ever', 'replace', 'ring', 'going', 'buy',
'hobby', 'one'], ['ordered', 'baby', 'blue', 'opal', 'cluster', 'king', 'loyal',
'customer', 'sterling', 'silver', 'blue', 'today', 'rings', 'upon', 'receiving',
'ring', 'totally', 'excited', 'anticipation', 'receiving', 'ring', 'exciting',
'particular', 'cluster', 'absolutely', 'beautiful', 'sparkling', 'color',
'stones', 'quality', 'ringreceived', 'numerous', 'compliments', 'would',
'recommend', 'ring', 'looking', 'ring', 'set', 'nicely', 'blue', 'today',
'stones', 'completely', 'satisfied', 'order', 'site', 'again'], ['sterling',
'silver', 'black', 'enamel', 'comes', 'life', 'cross', 'ring', 'five',
'crosses', 'stunning', 'ring', 'radiate', 'hand', 'many', 'comments', 'beauty',
'ring', 'grace', 'fills', 'one', 'soul', 'admire', 'beauty', 'interactions',
'silver', 'black', 'design', 'ring', 'ask', 'purchase', 'ring', 'share',
'inner', 'feelings', 'outward', 'way', 'without', 'say', 'work', 'allow',
'ring', 'speak', 'silver', 'black', 'enamel', 'cross', 'ring', 'speaks',
'selfutterly', 'undesirable'], ['fiance', 'looked', 'many', 'different',
'rings', 'fell', 'love', 'ring', 'everything', 'wanted', 'engagement', 'ring',
'wearing', 'awhile', 'perfect', 'regrets', 'buying', 'ring'], ['wanted', 'know',
'ring', 'like', '2', 'rings', 'one', 'ring', 'beyond', 'gorgeous', 'love'],
['bought', 'two', 'rings', 'fit', 'either', 'side', 'platino', 'princess',
'wedding', 'ring', 'help', 'anchor', 'add', 'ring', 'perfect', 'fit', 'lot',
'compliments', 'new', 'ring'], ['got', 'ring', 'birthday', 'love', 'not',
'imagine', 'woman', 'not', 'adoring', 'ring'], ['first', 'skeptically', 'ring',
'price', 'low', 'nt', 'wear', 'lot', 'gold', 'purchased', 'ring', 'nice',
'gold', 'necklaces', 'wear', 'wanted', 'ring', 'work', 'ring', 'nt', 'look',
'like', '35', 'ring'], ['ring', 'awesome', 'wear', 'ring', 'index', 'finger',
'fits', 'great', 'comfortable', 'ring'], ['not', 'ring', 'beautiful',
'jeweller', 'accommadating', 'ring', 'reach', 'us', 'time', 'appreciate',
'care', 'quick', 'receiving', 'ring', 'thank'], ['pleased', 'quality', 'item',
'definitely', 'recommend', 'addition', 'friends', 'family'], ['item',
'wonderful', 'surprise', 'quality', 'much', 'could', 'ever', 'hoped', 'for'],
['impressed', 'quality', 'item', 'delivery', 'fast', 'would', 'definitely',
'buy', 'seller', 'again'], ['impressed', 'quality', 'would', 'not', 'hesitate',
'purchase', 'items', 'seller', 'service', 'also', 'exceptional'], ['not',
'completely', 'sure', 'buying', 'jewellery', 'internet', 'type', 'gift', 'like',
'see', 'handle', 'item', 'decide', 'pressed', 'time', 'like', 'pictures',
'item', 'decided', 'take', 'chance', 'glad', 'impressed', 'quality',
'appearance', 'item', 'arrived', 'price', 'low', 'compared', 'quality', 'wife',
'pleased', 'jewellery', 'item'], ['flute', 'charm', 'detailed', 'high',
'quality', 'see', 'keys', 'flute', 'fan', 'would', 'adore', 'item'], ['stem',
'great', 'quality', 'came', 'promptly', 'happy', 'recommend', 'undeservedly'],
['see', 'person', 'appreciate', 'beautiful', 'really', 'shiny', 'feel',
'pretty', 'look', 'expensive', 'much', 'costthey', 'also', 'go', 'color',
'beautiful'], ['think', 'necklace', 'well', 'worth', 'money', 'delicate',
'simple', 'yet', 'pretty', 'layed', 'beautiful', 'look'], ['got', 'ring',
'promise', 'ring', 'girlfriend', 'christmas', 'loved', 'definitely', 'great',
'value'], ['wife', 'loves', 'ring', 'great', 'gift', 'extremely', 'cheap',
'high', 'quality'], ['birthday', 'gift', '16', 'niece', 'loves', 'ring',
'happy', 'received', 'it'], ['love', 'birthstone', 'wanted', 'piece', 'jewel',
'symbolized', 'simple', 'purity', 'blue', 'opal', 'ring', 'gift', 'birthday',
'year', 'definitely', 'great', 'gift', 'welcomed', 'addition', 'collection'],
['got', 'ring', 'gift', 'boyfriend', 'love', 'thing', 'rings', 'not',
'position', 'correctly', 'inches', 'skin'], ['ring', 'good', 'sparkle', 'looks',
'like', 'ring', 'cost', 'six', 'amount', 'takes', 'great', 'gift', 'someone',
'budget', 'girlfriend', 'loves', 'it'], ['great', 'gift', 'loved', 'one',
'ever', 'better', 'seller', 'seller', 'deals', 'professional', 'way',
'security', 'measures', 'superb'], ['bought', 'ring', 'husband', 'loved',
'received', 'said', 'would', 'great', 'ring'], ['product', 'made', 'great',
'gift', 'great', 'memories', 'love', 'something', 'always', 'helping', 'gift',
'heart', 'always', 'shows', 'care'], ['love', 'ring', 'fits', 'right', 'showed',
'daughter', 'ring', 'loved', 'well', 'great', 'everyday', 'wear', 'price',
'great'], ['nice', 'loves', 'birth', 'stone', 'got', 'christmas', 'lifti',
'also', 'love', 'also', 'great'], ['bought', 'gift', 'friends', 'birthday',
'loved', 'beautiful', 'ring'], ['always', 'love', 'pillow', 'free', 'make',
'great', 'gifts', 'great', 'people', 'life', 'quite', 'collection', 'hope',
'continue', 'build'], ['owned', 'several', 'claddagh', 'rings', 'years', 'lost',
'last', 'one', 'received', 'one', 'gift', 'recently', 'truly', 'love', 'sturdy',
'design', 'crisp', 'easily', 'recommend', 'one', 'lovely', 'ring'], ['mother',
'loved', 'great', 'birthday', 'gift', 'look', 'even', 'better', 'person', 'go',
'great', 'anything'], ['always', 'wanted', 'claddaugh', 'ringhis', 'price',
'greati', 'love', 'it'], ['disappointed', 'appearance', 'quality', 'bracelets',
'definitely', 'not', 'worth', '4500', 'not', 'even', 'close'], ['bracelets',
'not', 'true', '9', 'necklace', 'perfect', 'bracelets', 'nice', 'quality',
'not', 'true', 'length'], ['bought', 'bracelets', 'girlfriend', 'not',
'impressed', 'quality', 'bracelets', 'looks', 'cheap', 'clasp', 'extremely',
'fragile', 'fact', 'first', 'night', 'wore', 'bracelets', 'clasp', 'broke',
'one', 'side', 'definitely', 'would', 'not', 'recommend', 'bracelets'],
['bracelets', 'poor', 'quality', 'gemstones', 'poor', 'craftsmanship',
'gemstones', 'cloudy', 'joints', 'chain', 'fused', 'together', 'bracelets',
'therefore', 'ca', 'nt', 'straightened', 'completely', 'words', 'unfastened',
'bracelets', 'lay', 'counter', 'not', 'form', 'straight', 'line', 'joints',
'bracelets', 'fused', 'together', 'not', 'bend', 'not', 'straighten',
'bracelets', 'nt', 'buy', 'june'], ['stones', 'bracelets', 'extremely', 'pale',
'pink', 'purple', 'ended', 'returning', 'bracelets', 'amethyst', 'jewel',
'extremely', 'poor', 'quality'], ['purchased', 'bracelets', 'gift', 'last',
'december', 'quality', 'clasp', 'quite', 'poor', 'gradually', 'lost', 'ability',
'stay', 'closed', 'recently', 'bracelets', 'fell', 'poor', 'clasp', 'damaged',
'greatly', 'avoid', 'bracelets'], ['happy', 'product', 'received', 'advertised',
'timely', 'manner', 'selleramazon', 'kept', 'updated', 'shipmentdelivery',
'status', 'would', 'recommend', 'item', 'seller'], ['purchased', 'several',
'items', 'petya', 'not', 'products', 'first', 'rate', 'packing', 'customer',
'service', 'excellent'], ['earrings', 'described', 'transaction', 'smooth',
'easy', 'product', 'shipped', 'received', 'time', 'frame', 'quoted', 'pleased',
'purchase'], ['stem', 'shipped', 'received', 'within', 'time', 'limit', 'given',
'good', 'quality', 'product', 't'], ['wonderful', 'shopping', 'experience',
'purchased', 'item', 'holiday', 'presents', 'whole', 'order', 'came', 'quickly',
'wonderful', 'condition'], ['product', 'arrived', 'short', 'period', 'time',
'perfect', 'described', 'perfectly', 'everything', 'hoped'], ['received',
'italian', 'horn', 'printing', 'condition', 'completely', 'satisfied',
'receiving', 'product', 'timely', 'manner'], ['item', 'looks', 'exactly',
'like', 'pictures', 'pleased', 'shipping', 'ver', 'fast', 'item', '3', 'days',
'order', 'choosing', 'strange', 'shipping', 'worth', 'time'], ['arrived',
'estimated', 'date', 'many', 'previous', 'orders', 'seller', 'always', 'time',
'excellent', 'condition'], ['product', 'received', 'nice', 'came', 'timely',
'matter', 'faster', 'expected', 'order', 'item', 'again'], ['pleased',
'purchase', 'speedy', 'shipping', 'use'], ['happy', 'say', 'though', 'not',
'receive', 'order', 'first', 'time', 'ordered', 'cross', 'company', 'amazon',
'stand', 'behind', 'product', 'refund', 'money', 'spent', 'item', 'since',
'reordered', 'item', 'arrived', 'timely', 'manner', 'cross', 'really', 'nice',
'thank', 'great', 'customer', 'service'], ['product', 'arrived', 'perfect',
'condition', 'shipping', 'ridiculously', 'slow', 'not', 'order', 'again'],
['item', 'came', 'quickly', 'plenty', 'time', 'christmas', 'huge', 'hit',
'person', 'received', 'them'], ['bought', 'wear', 'alice', 'wonderland',
'costume', 'halloween', 'wearing', 'ever', 'since', 'received', 'actually',
'gotten', 'lot', 'compliments'], ['days', 'not', 'wear', 'blue', 'one', 'wear',
'one', 'really', 'enjoy', 'wearing', 'something', 'celtic', 'pretty'],
['peasant', 'classify', 'best', 'casual', 'wear', 'wear', 'weekend', 'nt',
'not', 'suited', 'work', 'going', 'events'], ['label', 'pin', 'perfect',
'detail', 'wear', 'colors', 'plan', 'wear', 'label', 'wear', 'suit', 'penis',
'nice', 'enough', 'wear', 'formal', 'occasionsdear', 'pride'], ['wanted',
'class', 'piece', 'wear', 'right', 'hand', 'work', 'wearing', 'old', 'found',
'end', 'wearing', 'outside', 'work', 'class', 'looking'], ['good', 'everyday',
'wear', 'dressing', 'up'], ['suitable', 'wearing', 'fashionable', 'occasions',
'dress'], ['looking', 'forward', 'wearing', 'sparkle', 'catch', 'every', 'eye',
'son', 'wedding', 'june'], ['wear', 'charm', 'memory', '3', 'friends', 'passed',
'away', 'wish', 'nt', 'wear', 'honor', 'do'], ['item', 'perfect', 'daily',
'wear', 'still', 'elegant', 'enough', 'dress', 'receive', 'many', 'compliments',
'wearing', 'it'], ['get', 'compliments', 'every', 'time', 'wear', 'necklace',
'quickly', 'become', 'part', 'everyday', 'wear'], ['great', 'way', 'support',
'local', 'pro', 'sports', 'team', 'without', 'wearing', 'oversized', 'jersey',
'hat', 'mess', 'hair'], ['nice', 'wear', 'want', 'something', 'casual', 'wear',
'comfortable'], ['unique', 'pleasure', 'wear', 'stones', 'catch', 'light',
'style', 'comfortable', 'wear'], ['item', 'not', 'pictured', 'funny', 'poor',
'quality', 'seller', 'not', 'respond', 'contracted', 'this'], ['product',
'hollow', 'not', 'clearly', 'specified', 'item', 'description', 'not',
'expecting', 'seemed', 'poor', 'quality', 'returned', 'item'], ['disappointed',
'quality', 'item', 'fragile', 'thin', 'discoloured', 'back', 'charm', 'not',
'returnable', 'not', 'ordering', 'amazon', 'jeweller', 'future', 'thank',
'you'], ['stem', 'arrived', 'extremely', 'damaged', 'several', 'places', 'not',
'package', 'well', 'send', 'back', 'disappointed', 'quality'], ['item',
'misrepresented', 'size', 'quality', 'horrible', 'would', 'return', 'item',
'except', 'family', 'member', 'coast', 'guard', 'sent', 'total', 'waste',
'money'], ['one', 'beautiful', 'rosary', 'seen', 'smoothness', 'color', 'beads',
'translucent', 'looking', 'almost', 'looks', 'like', 'glass', 'workmanship',
'excellent', 'details', 'beautiful', 'truly', 'beautiful', 'piece', 'own'],
['earrings', 'sent', 'real', 'light', 'color', 'not', 'pretty', 'dark', 'color',
'show', 'picture', 'look', 'almost', 'light', 'pink', 'keep', 'also', 'pretty',
'not', 'expected'], ['diamond', 'looks', 'pretty', 'big', 'price', 'shines',
'brilliantly', 'color', 'nt', 'look', 'white', 'though', 'nt', 'expect',
'color', 'white', 'overall', 'think', 'pretty', 'happy'], ['diamond', 'looks',
'pretty', 'big', 'price', 'shines', 'brilliantly', 'color', 'nt', 'look',
'white', 'though', 'nt', 'expect', 'color', 'white', 'overall', 'think',
'pretty', 'happy'], ['imagine', 'sparkling', 'around', 'girlfriend', 'manned',
'toe', 'sun', 'bad', 'winter', 'looking', 'forward', 'pretty', 'sight'],
['look', 'quite', 'like', 'photograph', 'colourful', 'know', 'turtle', 'seen',
'elsewhere', 'quite', 'high', 'price', 'beautiful'], ['ring', 'exactly',
'pictured', 'looks', 'pretty', 'hand', 'color', 'stones', 'rich', 'beautiful'],
['dainty', 'heart', 'looks', 'absolutely', 'beautiful', 'pick', 'colors',
'clothing', 'amazing', 'price', 'beautiful', 'peasant'], ['not', 'find',
'better', 'deal', 'money', 'anywhere', 'ring', 'beautiful', 'craftsmanship',
'meticulous', 'extensive', 'also', 'muscle', 'shells', 'tinker', 'person',
'look', 'online', 'deep', 'color', 'various', 'shades', 'iridescent', 'maude'],
['look', 'beautiful', 'smooth', 'finish', 'closes', 'firmly', 'click',
'classic', 'look'], ['smaller', 'carnelian', 'beads', 'clear', 'beautiful',
'orange', 'color', 'would', 'great', 'elastic', 'color', 'beads', 'beautiful'],
['message', 'positive', 'looks', 'pretty', 'bought', 'aunt', 'present', 'color',
'nice'], ['ring', 'looks', 'nothing', 'like', 'picture', 'diamonds', 'small',
'not', 'noticeable', 'sending', 'back'], ['ring', 'nice', 'looked', 'like',
'picture', 'crack', 'one', 'market', 'another', 'one', 'large', 'chip'],
['rings', 'looks', 'nothing', 'like', 'picture', 'stones', 'small', 'barely',
'even', 'tell', 'stones', 'ring', 'thin', 'small', 'looks', 'like', 'ring',
'would', 'buy', 'egg', 'machine', 'returned', 'promptly', 'misleading', 'buy'],
['nt', 'like', 'product', 'diamonds', 'looked', 'nothing', 'like', 'picture',
'diamonds', 'flowed', 'little', 'bit'], ['although', 'picture', 'shows', 'cut',
'looking', 'ring', 'ring', 'nt', 'pretty', 'fringed', 'look', 'looks', 'like',
'ring', 'left', 'floor', 'someone', 'ran', 'vacuum', 'cleaner'], ['usually',
'love', 'rings', 'company', 'warned', 'ring', 'nt', 'look', 'anything', 'like',
'picture', 'ring', 'received', 'super', 'long', 'rectangular', 'stone', 'not',
'nice', 'square', 'seen', 'picture', 'line', 'also', 'varnished', 'looked',
'like', 'torch', 'marks', 'side', 'definitely', 'returning', 'ring'], ['little',
'disappointed', 'received', 'ring', 'mail', 'picture', 'provided', 'sides',
'look', 'like', 'make', 'heart', 'shape', 'least', 'looks', 'like', 'smooth',
'clean', 'curved', 'lines', 'ring', 'got', 'mail', 'looks', 'like', 'sides',
'smashed', 'not', 'clean', 'curves', 'like', 'wished', 'looked', 'like',
'picture'], ['diamonds', 'picture', 'nothing', 'close', 'size', 'look',
'picture', 'ring', 'awesome', 'would', 'absolutely', 'recommend', 'anyone',
'price', 'hands', 'doubt', 'ring', 'looks', 'amazing', 'price', 'must', 'nt',
'expect', 'diamonds', 'look', 'big', 'picture'], ['received', 'ring', 'little',
'disappointed', 'ring', 'not', 'completely', 'blue', 'like', 'picture', 'shows',
'looks', 'like', 'got', 'blue', 'flower', 'green', 'leaves', 'makes', 'ring',
'look', 'blue', 'green', 'small', 'ring', 'not', 'worth', '699', 'like'],
['ring', 'looks', 'beautiful', 'picture', 'see', 'reality', 'cubic', 'zirconia',
'small', 'ring', 'looks', 'tiny', 'also', 'cubic', 'zirconia', 'dull', 'ring',
'received', 'looks', 'like', 'worn', 'disgusted', 'purchase', 'nt', 'buy',
'seller', 'not', 'like', 'false', 'imagine', 'ring', 'looks', 'cheap', 'sold',
'much', 'less', 'real', 'ripoff'], ['got', 'ring', 'hoping', 'looks', 'like',
'picture', 'misleading', 'purchase', 'ring', 'please', 'aware', '110', 'cut',
'means', 'diamonds', 'really', 'small', 'man', 'looking', 'buy', 'ring',
'significant', 'would', 'recommend', 'spend', 'little', 'bit', 'buy', 'better',
'looking', 'ring', 'ended', 'returning'], ['ring', 'clot', 'smaller', 'person',
'pictures', 'pictures', 'make', 'look', 'like', 'diamonds', 'decent', 'size',
'small', 'little', 'disappointed'], ['looks', 'like', 'ring', 'man', 'look',
'picture', 'online', 'real', 'life', 'feminine', 'looking', 'ring'], ['fell',
'love', 'picture', 'ring', 'showed', 'slightly', 'brushed', 'looking', 'ring',
'arrived', 'quick', 'learn', 'picture', 'looks', 'nothing', 'like', 'ring',
'ring', 'bright', 'polish', 'yellow', 'gold', 'barely', 'visible',
'disappointed', 'amazon', 'lack', 'description'], ['medical', 'alert',
'bracket', 'looked', 'like', 'picture', 'nice', 'quality', 'sterling',
'silver'], ['picture', 'looked', 'purple', 'clear', 'like', 'title', 'says'],
['earrings', 'picture', 'stones', 'look', 'good', 'light', 'comfortable',
'reasonable', 'quality', 'price', 'silver', 'not', 'look', 'picture', 'like',
'polished', 'silver'], ['much', 'smaller', 'looked', 'like', 'picture',
'silver', 'necklace', 'seemed', 'poorer', 'quality', 'expected'], ['nice',
'looks', 'picture', 'like'], ['perfect', 'size', 'solid', 'charm', 'looks',
'either', 'side', 'silver', 'nicely', 'finished', 'enamel', 'nice', 'highlight',
'really', 'looks', 'like', 'picture'], ['not', 'thought', 'going', 'look',
'good', 'picture', 'got', 'quality', 'nt', 'ca', 'nt', 'wear', 'completely',
'bending', 'hooks', 'shape', 'like', 'silver', 'plastic'], ['looked', 'well',
'picture', 'thing', 'could', 'say', 'little', 'polished', 'looks', 'like',
'black', 'stands', 'looks', 'nice'], ['diamond', 'crack', 'one', 'market',
'another', 'one', 'large', 'chip'], ['although', 'picture', 'looks', 'like',
'metal', 'beads', 'description', 'states', 'sterling', 'silver', 'pearls'],
['looks', 'exactly', 'like', 'picture', 'nice', 'quality', 'must', 'everyone',
'tiger', 'fan', 'owns', 'italian', 'harm', 'bracelets'], ['got', 'yesterday',
'christmas', 'gift', 'far', 'look', 'like', 'picture', 'seem', 'nice'], ['good',
'quality', 'light', 'weight', 'nice', 'small', 'size', 'must', 'described',
'look', 'like', 'picture'], ['really', 'liked', 'earrings', 'however', 'agree',
'one', 'earlier', 'reviews', 'thought', 'would', 'little', 'bigger'], []]
```

3 Question 2 - Information retrieval

[4]:
```
# Installations and imports for question 2
!pip install faiss-cpu
import numpy as np
from numpy import argsort
import scipy.sparse as sp
from sklearn.decomposition import TruncatedSVD
from sklearn.feature_extraction.text import CountVectorizer, TfidfTransformer,␣
 ↪TfidfVectorizer
from sklearn.metrics.pairwise import cosine_similarity
from sklearn.preprocessing import normalize
from sklearn import metrics
import torch
import torch.nn as nn
import transformers
from transformers import DistilBertModel, DistilBertTokenizerFast
from transformers import AdamW
import torch.nn.functional as F
import faiss
import warnings
warnings.filterwarnings('ignore')
```

```
Requirement already satisfied: faiss-cpu in /usr/local/lib/python3.11/dist-
packages (1.10.0)
Requirement already satisfied: numpy<3.0,>=1.25.0 in
/usr/local/lib/python3.11/dist-packages (from faiss-cpu) (1.26.4)
Requirement already satisfied: packaging in /usr/local/lib/python3.11/dist-
packages (from faiss-cpu) (24.2)
```

Evaluation functions. Used for both LSI and Neural Information Retrieval

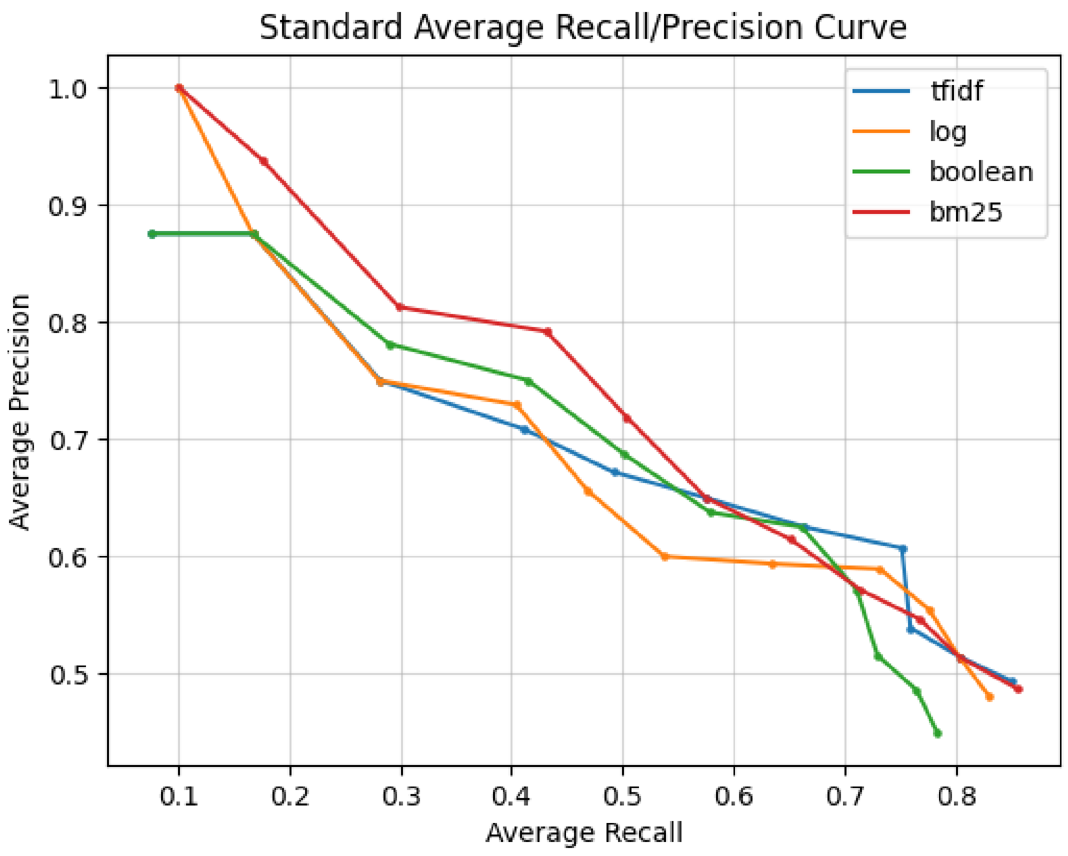
[5]:
```
# My own knowledge

import matplotlib.pyplot as plt

def calc_prec_rec_at_k(relevant_ids, retrieved_ids, k_values):
  '''Calculate precision and recall at different threshold values'''
  # Data stores
  prec_at_k = []
  rec_at_k = []
  # Edge case
  if len(relevant_ids) == 0:
    return [0] * len(k_values), [0] * len(k_values)

  # Calc precision and recall for different cut off (k values)
  for k in k_values:
    retrieved_ids_k = retrieved_ids[:k]
    tp_k = len(set(relevant_ids) & set(retrieved_ids_k))

    prec_at_k.append(tp_k / k)
    rec_at_k.append(tp_k / len(relevant_ids))

  return prec_at_k, rec_at_k

def plot_eval_curve(eval_dict):
  '''Plot precision-recall curve'''
  fig, ax = plt.subplots()

  for method in eval_dict.keys():
    ax.scatter(eval_dict[method][0], eval_dict[method][1], s=5)
    ax.plot(eval_dict[method][0], eval_dict[method][1], label = method)

  plt.legend()
  ax.grid(alpha=0.5)
  ax.set(xlabel='Average Recall', ylabel='Average Precision', title = 'Standard␣
↪Average Recall/Precision Curve')
```

3.1 Latent Semantic Indexing

[57]:
```
# The skeleton of this code was adapted from https://github.com/gcosma/COP509/
↪blob/main/Tutorials/Tutorial4LSA.ipynb

def get_vocab(clean_docs):
  '''Use sklearn Count Vectorizer to get vocabulary, using unigrams and bigrams␣
↪'''
  clean_docs = [' '.join(doc) for doc in clean_docs]
  vectorizer = CountVectorizer(ngram_range=(1,2), min_df=2, max_df=0.95)
  vectorizer.fit(clean_docs)
  return vectorizer.vocabulary_

def encode_data(train_docs, mode, vocab):
  '''Vectorize data then encode.
     Four different modes possible: tfidf, log, boolean, BM25'''
  vectorizer = CountVectorizer(vocabulary=vocab, ngram_range=(1,2))
  count_vectors = vectorizer.fit_transform(train_docs)
  if mode == "tfidf":
    # Tf-Idf encoding
    transformer = TfidfTransformer(norm='l2')
    Xtrain = transformer.fit_transform(count_vectors)

  elif mode == 'log':
    # Term freq
    Xtrain = np.log1p(count_vectors)
    # Document freq
    idf = sp.diags(TfidfTransformer(norm=None, use_idf=True).fit(count_vectors).
↪idf_,0)
    Xtrain=Xtrain.dot(idf)
    # Document length normalise
    Xtrain = normalize(Xtrain, norm='l2', axis=1)

  elif mode == 'boolean':
    # Term freq
    Xtrain = normalize((count_vectors > 0).astype(int), norm='l2', axis=1)
    # Document freq
    idf = sp.diags(TfidfTransformer(norm=None, use_idf=True).fit(count_vectors).
↪idf_,0)
    Xtrain=Xtrain.dot(idf)
    # Document length normalise
    Xtrain = normalize(Xtrain, norm='l2', axis=1)

  elif mode == 'bm25':
    # Using TfidfVectorizer with BM25 parameters
    bm25_vectorizer = TfidfVectorizer(vocabulary=vocab, ngram_range=(1,2),
                                  norm=None, use_idf=True,
                                  smooth_idf=False, sublinear_tf=True)
    Xtrain = bm25_vectorizer.fit_transform(train_docs)
  else:
    raise ValueError(f"Unsupported mode: {mode}. Use 'tfidf', 'log', 'boolean',␣
↪or 'bm25'")

  return Xtrain

def prepare_query(query, mode, vocab, encoded=True):
  '''Clean then encode query'''
  if len(query) == 2:
    query.pop(0) # dont need query number
  tokens = clean_doc(str(query))
  line = ' '.join(tokens)
  if encoded:
    encoded = encode_data([line], mode, vocab)
    return encoded
  else:
    return line

#Get vocab, read in necessary files
vocab = get_vocab(clean_text)
query_list = load_doc("QueryText.csv.xlsx")
query_relevant = pd.read_csv(data_path + "JewelleryReviewsQueryRelevantID.csv")
train_docs = [' '.join(doc) for doc in clean_text]

# Data stores
k_values = [1, 2, 4, 6, 8, 10, 12, 14, 16, 18, 20]
MAPs = []
method_avg_curves = {}

# Loop through each method, and query
# Calculate precision and recall at diff ks for eachs step
for method in ['tfidf', 'log', 'boolean', 'bm25']:
  lsi_precision_curves = []
  lsi_recall_curves = []

  # encode training data according to method
  Xtrain = encode_data(train_docs, method, vocab)
  trunc_SVD_model = TruncatedSVD(n_components=50)
  approx_Xtrain = trunc_SVD_model.fit_transform(Xtrain)

  for query, q in zip(query_list, range(0, len(query_list))):
    # prepare each query and query_vector in list according to method
    encoded_query = prepare_query(query, method, vocab)
    query_vector = trunc_SVD_model.transform(encoded_query)
    similarities = cosine_similarity(approx_Xtrain, query_vector)

    # sort by similarity
    indexes = np.argsort(similarities.flat)[::-1]
    retrieved_ids = [int(ids[i]) for i in indexes]

    query_relevant_ids = query_relevant.iloc[:,q].dropna().astype(int).tolist()

    # Calculate evaluation metrics
    prec_at_k, rec_at_k = calc_prec_rec_at_k(query_relevant_ids, retrieved_ids,␣
↪k_values)
    lsi_precision_curves.append(prec_at_k)
    lsi_recall_curves.append(rec_at_k)

  avg_precision_curve = np.mean(lsi_precision_curves, axis=0)
  avg_recall_curve = np.mean(lsi_recall_curves, axis=0)
  method_avg_curves[method] = (avg_recall_curve, avg_precision_curve)
  MAPs.append(sum(avg_precision_curve)/len(avg_precision_curve))
```

```
# Output results
plot_eval_curve(method_avg_curves)

for i, method in enumerate(method_avg_curves.keys()):
  print(f'Mean avg precision {method}: {MAPs[i]:.2f}')

for method in method_avg_curves.keys():
  print(f'Area under precision/recall curve for method {method}: {metrics.
↪auc(method_avg_curves[method][0], method_avg_curves[method][1]):.2f}')
```

```
Mean avg precision tfidf: 0.66
Mean avg precision log: 0.67
Mean avg precision boolean: 0.66
Mean avg precision bm25: 0.69
Area under precision/recall curve for method tfidf: 0.54
Area under precision/recall curve for method log: 0.50
Area under precision/recall curve for method boolean: 0.51
Area under precision/recall curve for method bm25: 0.55
```



[7]:
```
print(vocab)
```

{'expect': 204, 'like': 311, 'regular': 504, 'size': 621, 'ring': 515, 'one':
412, 'look': 324, 'toy': 690, 'something': 638, 'funny': 238, 'rings': 569,
'may': 365, 'diamond': 162, 'king': 301, 'ask': 39, 'white': 728, 'old': 411,
'ring one': 549, 'look like': 327, 'like ring': 316, 'ring may': 544, 'ordered':
422, 'toe': 684, 'description': 159, 'came': 89, 'quickly': 486, 'not': 393,
'nt': 405, 'know': 302, 'anyone': 31, 'please': 453, 'seller': 591, 'wedding':
724, 'band': 47, 'thumb': 677, 'ordered ring': 423, 'ring came': 525, 'came
quickly': 90, 'toe ring': 685, 'ring nt': 548, 'size toe': 622, 'wedding band':
725, 'ring thumb': 565, 'thumb ring': 678, 'beautiful': 51, 'first': 223,
'shipment': 601, 'scratches': 584, 'missing': 372, 'diamonds': 166, 'shoulder':
607, 'returned': 508, 'replacement': 506, 'second': 586, 'even': 189, 'worse':
745, 'clearly': 105, 'visible': 706, 'scratch': 581, 'right': 513, 'middle':
368, 'black': 68, 'great': 256, 'time': 679, 'refund': 502, 'thanks': 663,
'clot': 108, 'amazon': 24, 'ring beautiful': 521, 'beautiful ring': 54, 'ring
first': 533, 'first shipment': 224, 'shipment ring': 602, 'ring scratches': 559,
'scratches diamond': 585, 'diamond missing': 164, 'missing diamonds': 373,
'diamonds shoulder': 167, 'shoulder ring': 608, 'ring returned': 556, 'returned
ring': 510, 'ring replacement': 555, 'replacement ring': 507, 'ring second':
560, 'second ring': 587, 'ring even': 530, 'even worse': 193, 'worse clearly':
746, 'clearly visible': 106, 'visible white': 707, 'white scratch': 730,
'scratch right': 582, 'right middle': 514, 'middle black': 369, 'black diamond':
69, 'diamond not': 165, 'not look': 398, 'look great': 326, 'great time': 260,
'time diamond': 681, 'missing shoulder': 374, 'returned refund': 509, 'refund
thanks': 503, 'thanks clot': 664, 'clot amazon': 109, 'huge': 277, 'waste': 712,
'money': 375, 'month': 376, 'part': 430, 'holds': 274, 'symbol': 660,
'necklace': 379, 'broke': 80, 'low': 355, 'quality': 481, 'item': 291, 'waste
money': 713, 'quality item': 483, 'would': 749, 'recommend': 498, 'would not':
752, 'not recommend': 399, 'recommend item': 500, 'top': 688, 'scratched': 583,
'line': 319, 'saw': 579, 'store': 652, 'purchased': 478, 'item not': 293,
'hard': 266, 'ca': 87, 'wear': 716, 'easy': 178, 'got': 254, 'ca nt': 88, 'nt
wear': 410, 'nt like': 408, 'wanted': 709, 'beyond': 60, 'gorgeous': 251,
'love': 347, 'wanted know': 710, 'know ring': 303, 'ring like': 540, 'like
rings': 317, 'rings one': 574, 'one ring': 416, 'ring beyond': 522, 'beyond
gorgeous': 61, 'gorgeous love': 252, 'expensive': 208, 'stone': 650, 'worth':
747, 'returning': 511, 'cost': 130, 'not worth': 401, 'ring cost': 527, 'wore':
741, 'sits': 620, 'broken': 81, 'sterling': 645, 'silver': 614, 'things': 669,
'sterling silver': 646, 'silver rings': 615, 'way': 714, 'small': 628, 'looks':
335, 'want': 708, 'nice': 387, 'small looks': 629, 'looks like': 339, 'fell':
218, 'metal': 367, 'already': 17, 'spend': 643, 'favorite': 217, 'mine': 370,
'cut': 143, 'seemed': 589, 'lot': 344, 'prettier': 460, 'sparkling': 642,
'turned': 695, 'since': 618, 'compliment': 125, 'anything': 32, 'sized': 623,
'stick': 647, 'different': 168, 'nice small': 390, 'small sized': 632, 'sized
ring': 625, 'ring stick': 563, 'stick rings': 648, 'rings different': 570,
'different looks': 169, 'birthday': 64, 'imagine': 279, 'woman': 738, 'adoring':
11, 'got ring': 255, 'ring birthday': 523, 'birthday love': 66, 'love not': 348,
'not imagine': 397, 'imagine woman': 280, 'woman not': 739, 'not adoring': 394,
'adoring ring': 12, 'skeptically': 626, 'price': 464, 'gold': 246, 'necklaces':
380, 'work': 742, '35': 1, 'first skeptically': 225, 'skeptically ring': 627,

```
'ring price': 552, 'price low': 466, 'low nt': 356, 'wear lot': 717, 'lot gold':
346, 'gold purchased': 248, 'purchased ring': 479, 'ring nice': 546, 'nice
gold': 388, 'gold necklaces': 247, 'necklaces wear': 381, 'wear wanted': 721,
'wanted ring': 711, 'ring work': 567, 'work ring': 743, 'nt look': 409, 'like
35': 312, '35 ring': 2, 'awesome': 44, 'index': 285, 'finger': 221, 'fits': 229,
'comfortable': 117, 'ring awesome': 520, 'awesome wear': 45, 'wear ring': 719,
'ring index': 539, 'index finger': 286, 'finger fits': 222, 'fits great': 230,
'great comfortable': 257, 'comfortable ring': 118, 'jeweller': 297,
'accommadating': 5, 'reach': 488, 'us': 701, 'appreciate': 34, 'care': 91,
'quick': 484, 'receiving': 495, 'thank': 662, 'not ring': 400, 'beautiful
jeweller': 52, 'jeweller accommadating': 298, 'accommadating ring': 6, 'ring
reach': 553, 'reach us': 489, 'us time': 702, 'time appreciate': 680,
'appreciate care': 35, 'care quick': 92, 'quick receiving': 485, 'receiving
ring': 496, 'ring thank': 564, 'everyday': 197, 'get': 239, 'me': 366, 'love
ring': 349, 'ring not': 547, 'ring would': 568, 'bought': 72, 'gift': 240,
'could': 131, 'hands': 264, 'heart': 269, 'kept': 300, 'bought ring': 73, 'could
nt': 132, 'cut ring': 144, 'little': 320, 'better': 57, 'pink': 449, 'ring
little': 541, 'small ring': 630, 'ring finger': 532, 'better pink': 59, 'pink
ring': 450, 'pretty': 461, 'enough': 187, 'easily': 177, 'ring pretty': 551,
'husband': 278, 'loves': 353, 'thing': 668, 'made': 357, 'good': 249,
'definitely': 154, 'help': 270, 'however': 276, 'wearing': 722, 'next': 386,
'days': 150, 'tried': 691, 'notice': 404, 'cool': 129, 'complaint': 122,
'yellow': 756, 'much': 377, 'overall': 427, 'happy': 265, 'ring complaint': 526,
'rings not': 573, 'similar': 616, 'site': 619, 'ended': 183, 'back': 46,
'really': 491, 'wish': 734, 'around': 36, 'way around': 715, 'around ring': 37,
'ring great': 537, 'shipped': 603, 'shipping': 605, 'especially': 188, 'wears':
723, '10': 0, 'new': 384, 'guess': 262, 'order': 421, 'online': 418, 'quite':
487, 'see': 588, 'comfortable wear': 119, 'reviews': 512, 'picture': 437,
'pictures': 447, 'two': 696, 'going': 245, 'received': 493, 'smaller': 634,
'cross': 138, 'design': 160, 'different rings': 170, 'ring received': 554,
'shows': 610, 'blue': 70, 'disappointed': 171, 'dark': 147, 'stones': 651,
'advertised': 13, 'ring definitely': 529, 'definitely not': 156, 'also': 18,
'ever': 194, 'buying': 86, 'bit': 67, 'bulkier': 82, 'last': 306, 'liked': 318,
'catches': 96, 'attention': 40, 'ring also': 517, 'one first': 413, 'product':
471, 'fast': 216, 'small rings': 631, 'rings ring': 575, 'attractive': 41,
'high': 272, 'high quality': 273, 'rosary': 577, 'pleased': 454, 'opened': 420,
'need': 382, 'aware': 42, 'present': 459, 'sure': 658, 'fact': 212, 'looking':
332, 'items': 294, 'items seller': 295, 'still': 649, 'took': 687, 'less': 308,
'almost': 15, 'impossible': 281, 'wear necklace': 718, 'almost impossible': 16,
'sparkle': 640, 'dainty': 145, 'must': 378, 'for': 234, 'expecting': 207,
'color': 111, 'barely': 49, 'thought': 676, 'buy': 83, 'piece': 448, 'looked':
330, 'long': 323, 'replace': 505, 'hand': 263, 'given': 242, 'perfect': 433,
'delicate': 157, 'sturdy': 654, 'lovely': 352, 'detail': 161, 'elegant': 181,
'beautiful piece': 53, 'fit': 226, 'either': 179, 'side': 611, 'platino': 451,
'princess': 469, 'anchor': 26, 'add': 8, 'compliments': 126, 'bought two': 74,
'two rings': 697, 'rings fit': 572, 'fit either': 227, 'either side': 180, 'side
platino': 613, 'platino princess': 452, 'princess wedding': 470, 'wedding ring':
726, 'ring help': 538, 'help anchor': 271, 'anchor add': 27, 'add ring': 9,
'ring perfect': 550, 'perfect fit': 434, 'fit lot': 228, 'lot compliments': 345,
'compliments new': 127, 'new ring': 385, 'say': 580, 'shines': 598, 'perfectly':
435, 'ring shines': 561, 'addition': 10, 'wonderful': 740, 'sending': 593,
'daughter': 149, 'ring ring': 558, 'girlfriend': 241, 'thickness': 665,
'always': 21, 'solid': 636, 'thickness ring': 666, 'absolutely': 3,
'engagement': 185, 'unable': 698, 'ring got': 536, 'engagement ring': 186, 'go':
244, 'find': 220, 'use': 703, 'ring fits': 534, 'nt buy': 406, 'celtic': 98,
'worried': 744, 'wide': 732, 'though': 674, 'man': 360, 'big': 62, 'arrived':
38, 'company': 121, 'fits right': 232, 'band since': 48, 'even though': 192,
'ring bought': 524, 'would definitely': 750, 'everyone': 199, 'comments': 120,
'would recommend': 753, 'recommend ring': 501, 'real': 490, 'friends': 237,
'cubic': 139, 'worth money': 748, 'owned': 429, 'far': 215, 'individual': 287,
'yet': 757, 'individual rings': 288, 'flower': 233, 'love rings': 350, 'date':
148, 'this': 673, 'actually': 7, 'unfortunately': 699, 'purchase': 475,
'without': 736, 'vacation': 704, 'otherwise': 425, 'harm': 267, 'dont': 173,
'unfortunately small': 700, 'small woman': 633, 'ring size': 562, 'impossible
purchase': 282, 'purchase rings': 477, 'rings without': 576, 'without sized':
737, 'sized purchased': 624, 'would something': 754, 'something wear': 639,
'wear vacation': 720, 'vacation things': 705, 'things could': 670, 'could
otherwise': 133, 'otherwise harm': 426, 'harm expensive': 268, 'expensive
rings': 209, 'rings dont': 571, 'dont even': 174, 'even need': 191, 'need
expensive': 383, 'fits looks': 231, 'looks beautiful': 336, 'stunning': 653,
'amethyst': 25, 'market': 363, 'prices': 468, 'ring absolutely': 516, 'ring
anyone': 519, 'recently': 497, 'lost': 343, 'another': 28, 'sparkles': 641,
'ring much': 545, 'like one': 313, 'another ring': 30, 'ring look': 542,
'recommend anyone': 499, 'turn': 694, 'green': 261, 'surprise': 659, 'together':
686, 'make': 358, 'looked like': 331, 'ring always': 518, 'suggest': 656,
'every': 195, 'jewel': 296, 'collection': 110, 'wife': 733, 'course': 134,
'zirconia': 758, 'believe': 56, 'truly': 693, 'ring wear': 566, 'not expecting':
396, 'ring course': 528, 'cubic zirconia': 140, 'makes': 359, 'glad': 243,
'decided': 153, 'ordering': 424, 'year': 755, 'seen': 590, 'style': 655, 'eye':
211, 'catching': 97, 'cheap': 100, 'beautifully': 55, 'losing': 342, 'think':
671, 'deal': 151, 'fan': 214, 'ring right': 557, 'little bit': 321, 'tell': 661,
'bought wear': 75, 'every time': 196, 'exactly': 201, 'ring exactly': 531,
'great ring': 259, 'night': 392, 'gorgeous ring': 253, 'december': 152, 'box':
76, 'away': 43, 'suit': 657, 'excellent': 203, 'last december': 307, 'one wear':
417, 'expected': 206, 'totally': 689, 'true': 692, 'person': 436, 'shopping':
606, 'people': 432, 'picture looked': 440, 'even better': 190, 'better person':
58, 'catch': 95, 'thin': 667, 'within': 735, 'timely': 682, 'manner': 361,
'timely manner': 683, 'light': 310, 'shiny': 600, 'buy ring': 84, 'although':
19, 'opal': 419, 'customer': 141, 'nicely': 391, 'completely': 123, 'satisfied':
578, 'again': 14, 'blue opal': 71, 'absolutely beautiful': 4, 'color stones':
114, 'looking ring': 334, 'completely satisfied': 124, 'enamel': 182, 'life':
309, 'many': 362, 'purchase ring': 476, 'everything': 200, 'fell love': 219,
'family': 213, 'hoped': 275, 'impressed': 283, 'impressed quality': 284, 'buy
seller': 85, 'service': 595, 'appearance': 33, 'not completely': 395, 'like
pictures': 315, 'item arrived': 292, 'charm': 99, 'stem': 644, 'promptly': 474,
'well': 727, 'simple': 617, 'christmas': 102, 'loved': 351, 'definitely great':
```

```
155, 'extremely': 210, 'loves ring': 354, 'great gift': 258, 'it': 289,
'birthday gift': 65, 'ring gift': 535, 'someone': 637, 'would great': 751,
'showed': 609, 'everyday wear': 198, 'several': 596, 'bracelets': 77, 'close':
107, 'quality bracelets': 482, 'nice quality': 389, 'clasp': 103, 'fragile':
236, 'looks cheap': 337, 'side definitely': 612, 'poor': 457, 'craftsmanship':
137, 'june': 299, 'poor quality': 458, 'purple': 480, 'ended returning': 184,
'damaged': 146, 'product received': 473, 'customer service': 142, 'earrings':
176, 'described': 158, 'smooth': 635, 'shipped received': 604, 'pleased
purchase': 455, 'good quality': 250, 'condition': 128, 'product arrived': 472,
'italian': 290, 'looks exactly': 338, 'exactly like': 202, 'receive': 492,
'peasant': 431, 'casual': 93, 'casual wear': 94, 'colors': 116, 'dress': 175,
'forward': 235, 'looking forward': 333, 'pictured': 446, 'seller not': 592,
'disappointed quality': 172, 'sent': 594, 'beads': 50, 'color beads': 112,
'brilliantly': 78, 'diamond looks': 163, 'looks pretty': 341, 'pretty big': 462,
'big price': 63, 'price shines': 467, 'shines brilliantly': 599, 'brilliantly
color': 79, 'color nt': 113, 'look white': 329, 'white though': 731, 'though
nt': 675, 'nt expect': 407, 'expect color': 205, 'color white': 115, 'white
overall': 729, 'overall think': 428, 'think pretty': 672, 'pretty happy': 463,
'price beautiful': 465, 'amazing': 22, 'amazing price': 23, 'clear': 104,
'nothing': 402, 'ring looks': 543, 'looks nothing': 340, 'nothing like': 403,
'like picture': 314, 'picture diamonds': 438, 'crack': 135, 'large': 304,
'chip': 101, 'crack one': 136, 'one market': 415, 'market another': 364,
'another one': 29, 'one large': 414, 'large chip': 305, 'misleading': 371,
'picture stones': 445, 'although picture': 20, 'picture shows': 444, 'picture
ring': 443, 'shape': 597, 'little disappointed': 322, 'received ring': 494,
'look picture': 328, 'picture looks': 441, 'picture nice': 442, 'polished': 456,
'look good': 325, 'picture like': 439}
```

3.1.1 LSI comments

From the graph we can see that all four methods performed comparably, with BM25 having the best balance between precision and recall with the largest area on the curve at 0.55, and a mean avg precision of 0.69 but not by a lot.

Precision starts high (around 1.0, when k = 1 some methods are able to retrieve the correct relevant document) but decreases to approximately 0.5 once k = 20.

3.2 Neural Information Retreival

Prepare the dataset for training

- Extract the ids [8x193]
- Extract the relevant/not relevant labels for each query [8x193]
- Using the ids to index the location of the doc in `clean_text` extract the docs [8x193]

This produces an easily indexible list of lists, with index 0 corresponding to ids, relevancy labels and docs for query 1 and so on

[8]:
```
# My own knowledge
```

```
pos_ids = [query_relevant[query].dropna().astype(int).to_list() for query in␣
 ↪['Query1', 'Query2', 'Query3', 'Query4', 'Query5', 'Query6', 'Query7',␣
 ↪'Query8']]

ir_ids =[[] for _ in range(8)]
ir_labels= [[] for _ in range(8)]
for i, query in enumerate(['Query1', 'Query2', 'Query3', 'Query4', 'Query5',␣
 ↪'Query6', 'Query7', 'Query8']):
  # Relevant ids for each query
  rels = pos_ids[i]
  # Loop through all ids
  for id in ids:
    # Append to id list, and if relevant append 1 to labels
    id = int(id)
    if id in rels:
      ir_ids[i].append(id)
      ir_labels[i].append(1)
    else:
      ir_ids[i].append(id)
      ir_labels[i].append(0)

# Now want to get the actual texts that correspond to the ids in the pos and␣
 ↪neg lists
ir_idxs = [[] for _ in range(8)]
ir_text = [[] for _ in range(8)]
for i in range(8):
  # Get the location of each relevant id
  ir_idxs[i].append([ids.index(str(id)) for id in ir_ids[i]])
  # Use that to index the documents
  ir_text[i].extend([clean_text[idx] for idx in ir_idxs[i][0]])
```

Train the dataset. Due to time, data and memory constraints, this model was only trained using one batch of ten data points for arbitrarily chosen query 7 - so it probably didn't particularly help the model performance - but good practise

[9]:
```
# This code was adapted from https://huggingface.co/transformers/v3.0.2/
 ↪training.html

def loss(query_emb, emb, labels, margin=0.2):
  '''Use the Cosine embedding loss function '''
  # Extract embeddings (CLS token)
  query_emb = query_emb.last_hidden_state[:, 0, :]
  emb = emb.last_hidden_state[:, 0, :]

  # Get target labels for CosineEmbeddingLoss (-1 for negative, 1 for positive)
  target = torch.where(labels == 1, torch.tensor(1.0), torch.tensor(-1.0))
```

```
    # CosineEmbeddingLoss
    loss_fn = nn.CosineEmbeddingLoss(margin=margin)

    # Calculate loss
    loss = loss_fn(query_emb, emb, target)
    return loss

# Load the Pre-trained distilBERT model
checkpoint = 'distilbert-base-uncased'
tokenizer = DistilBertTokenizerFast.from_pretrained(checkpoint)
model = DistilBertModel.from_pretrained(checkpoint)

# Init optimiser
optimizer = AdamW(model.parameters(), lr=1e-5)

# Data - chose query 7 arbitrarily for training
text_batch = [' '.join(t) for t in ir_text[6]]
labels = ir_labels[6]
query = query_list[6]

# Batch data (time and mem constraints, would normally use a larger proportion)
text_batch = text_batch[159:169] #include some relevant docs as well
labels = labels[159:169]

# Tokenize data
t_encoding = tokenizer(text_batch, return_tensors='pt', padding=True,␣
 ↪truncation=True)
q_encoding = tokenizer(query, return_tensors='pt', padding=True,␣
 ↪truncation=True)

# Forward pass over the model
emb = model(**t_encoding)
quer_emb = model(**q_encoding)

# Train model
loss = loss(quer_emb, emb, torch.tensor(labels))
loss.backward()
optimizer.step()

# Save model
torch.save(model.state_dict(), "distilbert_retrieval.pth")
```

Load in the model

```
[10]: model_bert = DistilBertModel.from_pretrained(checkpoint)
model_bert.load_state_dict(torch.load("distilbert_retrieval.pth",␣
 ↪weights_only=True))
```

```
[10]: <All keys matched successfully>
```

Now ready to do the actual retrieval:

```
[58]: # embed_text function https://github.com/gcosma/COP509/blob/main/Tutorials/
       ↪Tutorial6UsingBERTfortheFirstTime.ipynb
      # faiss https://github.com/facebookresearch/faiss/wiki/getting-started
      # Method https://medium.com/@mhammadkhan/neural-re-ranking-models-c0a67278f626



      def embed_text(texts, max_length=512):
          '''Use the trained bert model to embed texts - get the CLS token '''
          # Tokenize text and pad
          tokenized = tokenizer(texts, add_special_tokens=True, padding=True,
                                truncation=True, max_length=max_length,␣
       ↪return_tensors="pt")
          # Get embeddings
          with torch.no_grad():
              last_hidden_states = model_bert(**tokenized)
          embeddings = last_hidden_states[0][:, 0, :].numpy()

          return embeddings

      def build_faiss_index(embeddings):
          '''Build a faiss index to store the embeddings, its faster for lookups '''
          index = faiss.IndexFlatL2(embeddings.shape[1])
          index.add(embeddings)
          return index

      def retreive(query, index, top_k):
          '''Retrieve top_k docs by their ids'''
          # Embed query and search the faiss index for close docs
          query_embedding = embed_text(query)
          distances, indexes = index.search(query_embedding, top_k)
          return [int(ids[i]) for i in indexes[0]]

      def rerank(query, retrieved_ids, top_k=10):
          '''Re rank retrieved docs using the cosine distance '''
          # Embed query and retrieved docs
          query_embedding = embed_text(query)  # Embed the query

          retrieved_idxs = np.where(np.isin(ids, retrieved_ids))[0]  # Get indexes of␣
       ↪retrieved doc ids
          retrieved_texts = [texts[i] for i in retrieved_idxs] # Get the retrieved docs
          retrieved_texts = [' '.join(t) for t in texts] # format
```

```
    retrieved_embeddings = embed_text(retrieved_texts)  # Embed the retrieved␣
 ↪documents

    # Compute cosine similarity between query and each retrieved document
    similarities = cosine_similarity(query_embedding, retrieved_embeddings)

    # Get the indices of the retrieved docs based on similarity
    reranked_idxs = np.argsort(similarities[0])[::-1]
    reranked_ids = [ids[i] for i in reranked_idxs]

    return reranked_ids

# Prepare texts, embed them, and store the embeddings in the faiss index
texts = [' '.join(t) for t in text]
embeddings = embed_text(texts)
faiss_index = build_faiss_index(embeddings)

# Data stores
precision=[]
recall=[]

# Loop through each query
for query, q in zip(query_list, range(0, len(query_list))):
    #Retrieve and re rank ids
    retrieved_ids = retreive(query, faiss_index, 30)
    reranked_ids = rerank(query, retrieved_ids)

    query_relevant_ids = query_relevant.iloc[:,q].dropna().astype(int).tolist()

    # Calculate Evaluation metrics
    prec_at_k, rec_at_k = calc_prec_rec_at_k(query_relevant_ids, retrieved_ids,␣
 ↪k_values)
    precision.append(prec_at_k)
    recall.append(rec_at_k)

avg_precision = np.mean(precision, axis=0)
avg_recall = np.mean(recall, axis=0)
MAP = (sum(avg_precision)/len(avg_precision))

# Output reuslts
plot_eval_curve({'NIR': (avg_recall, avg_precision)})

print(f'Mean avg precision {method}: {MAP:.2f}')
print(f'Area under precision/recall curve for method DistilBert NIR: {metrics.
 ↪auc(avg_recall, avg_precision):.2f}')
```

3.2.1 Neural Information Retrieval model comments

Evidently the LSI model performed much better. The model was only trained on ten data points for question 7 making it unlikely to be able to generalise well. The neural retrieval precision/recall graph has a much lower precision

4 Question 3

4.1 Clustering

4.1.1 Imports for clustering and relevant functions

[59]:
```
from sklearn.cluster import KMeans, AgglomerativeClustering
from sklearn.decomposition import TruncatedSVD, PCA
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import silhouette_score, davies_bouldin_score
from scipy.spatial.distance import cdist
from sklearn.feature_selection import SelectKBest, f_classif
import numpy as np
import pandas as pd

def closest_docs_to_centroid(text_vecs, centroids, labels):
  #calc distances from each doc to each centroid
  distances = cdist(text_vecs, centroids, 'cosine')

  closest_docs=[]
  for i in range(len(centroids)):
    cluster_docs_idx = np.where(labels == i)[0]

    closest_doc_idx = np.argsort(distances[cluster_docs_idx, i])
    sorted_closest_docs = cluster_docs_idx[closest_doc_idx]
    closest_docs.append(sorted_closest_docs[:5])

  return closest_docs

def cluster_wrapper(method, data, n_clusters):
  if method == 'agglom':
    clustering = AgglomerativeClustering(n_clusters=n_clusters,␣
↪metric='cosine', linkage='average').fit(data)
    return clustering
  elif method == 'kmeans':
    kmeans = KMeans(n_clusters=n_clusters, random_state=0).fit(data)
    return kmeans
```

4.1.2 We know from the graph that LSI with BM25 encoding works the best - so lets use that - with arbitrarily chosen query 3

This is just a repeat of the above LSI code without the loops and metric calculations

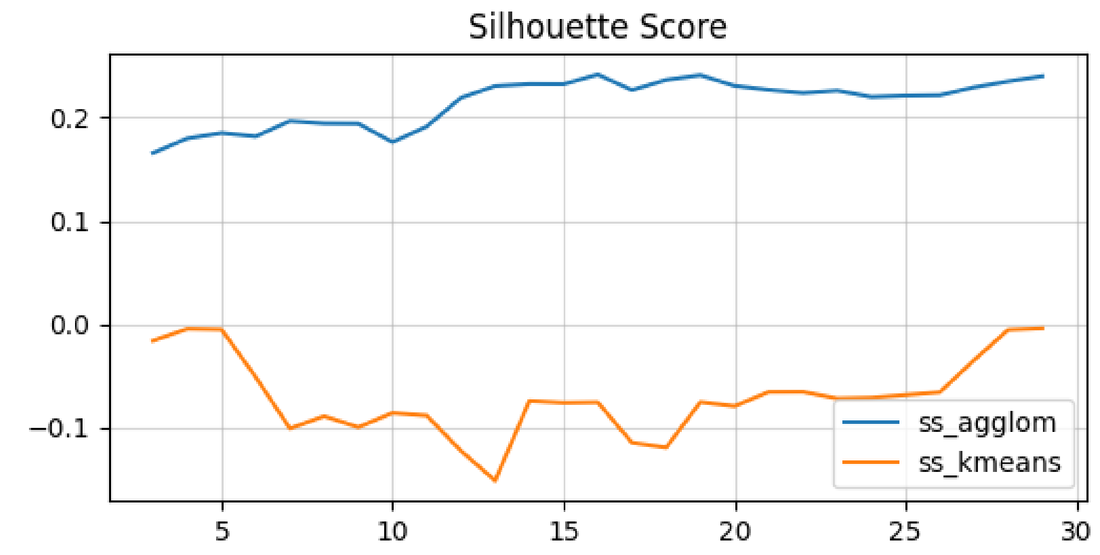
```
[60]: # As stated above, adapted from https://github.com/gcosma/COP509/blob/main/
      ↪Tutorials/Tutorial4LSA.ipynb

      Xtrain = encode_data(train_docs, 'bm25', vocab)
      trunc_SVD_model = TruncatedSVD(n_components=50)
      approx_Xtrain = trunc_SVD_model.fit_transform(Xtrain)

      # prepare each query and query_vector in list according to method
      encoded_query = prepare_query(query_list[2], 'bm25', vocab)
      query_vector = trunc_SVD_model.transform(encoded_query)
      similarities = cosine_similarity(approx_Xtrain, query_vector)

      # sort by similarity
      indexes = np.argsort(similarities.flat)[::-1]
      retrieved_ids = [int(ids[i]) for i in indexes]

      #store appropriate retrieved doc vectors
      retrieved = [approx_Xtrain[i] for i in indexes]
```

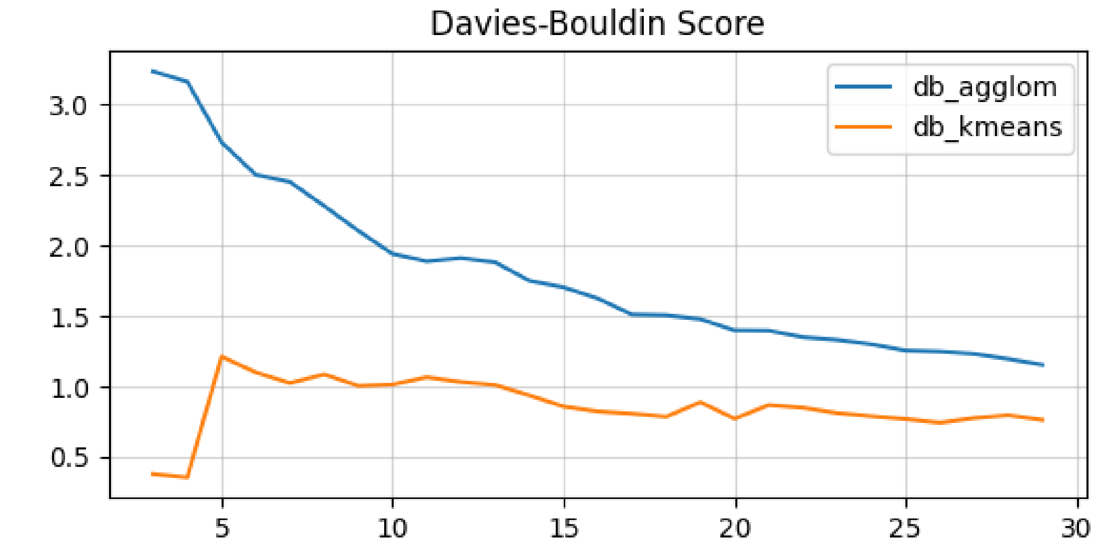
4.1.3 Now ready to start clustering

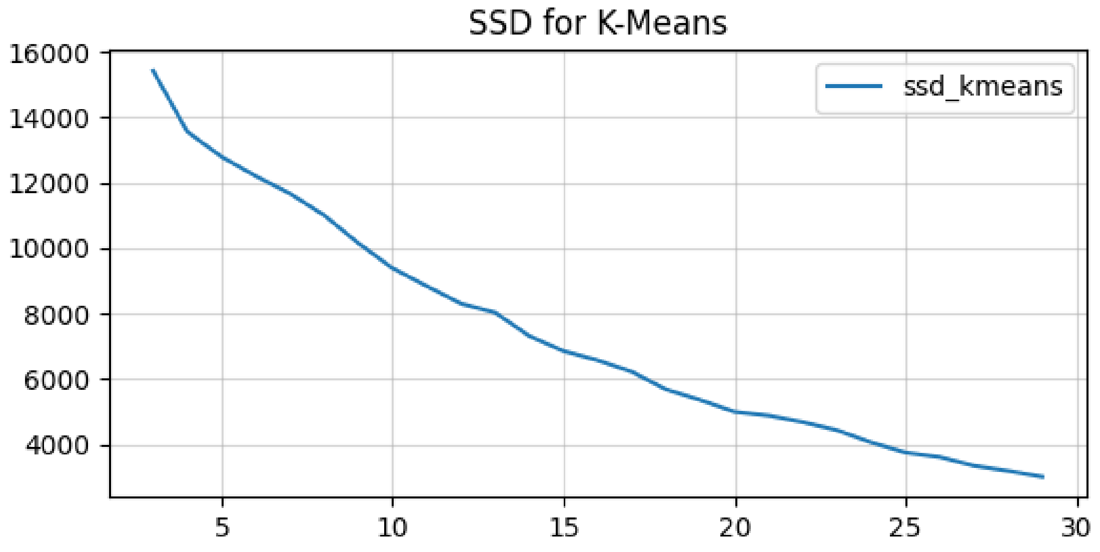
```
[61]: # My own knowledge with help from docs https://scikit-learn.org/stable/

      #chosen query, lsi with bm25 normalisation
      top_100 = retrieved[:100]
      top_100_ids = retrieved_ids[:100]
      n_clus=30

      #scale input data
      scaler = MinMaxScaler()
      top_100_sc = scaler.fit_transform(top_100)

      ss_score=[[],[]]
      db_score=[[],[]]
      ssd_score=[[],[]]

      #calculate clusters, store results
      for method in ['agglom', 'kmeans']:
        for n_clusters in range(3,n_clus):
          clusters = cluster_wrapper(method, top_100_sc, n_clusters)
          #select best features
          selector = SelectKBest(f_classif, k=1000)
          X_selected = selector.fit_transform(top_100, clusters.labels_)
          #re cluster
          clusters = cluster_wrapper(method, X_selected, n_clusters)

          if method == 'agglom':
```

```
      ss_score[0].append(silhouette_score(top_100, clusters.labels_,␣
 ↪metric='cosine'))
      db_score[0].append(davies_bouldin_score(top_100, clusters.labels_))

    else:
      ss_score[1].append(silhouette_score(top_100, clusters.labels_,␣
 ↪metric='cosine'))
      db_score[1].append(davies_bouldin_score(top_100, clusters.labels_))
      ssd_score[1].append(np.sum(clusters.inertia_))

cluster_scores = pd.DataFrame({'n_clusters':range(3,n_clus),
                               'ss_agglom':ss_score[0], 'ss_kmeans':ss_score[1],
                               'db_agglom':db_score[0], 'db_kmeans':db_score[1],
                               'ssd_kmeans':ssd_score[1]})
```

[62]:
```
# My own knowledge

# Plot each metric
fig, ax = plt.subplots(3, 1, figsize=(6,9))
metrics = [('ss_agglom', 'ss_kmeans'), ('db_agglom', 'db_kmeans'),␣
 ↪('ssd_kmeans', None)]
titles = ['Silhouette Score', 'Davies-Bouldin Score', 'SSD for K-Means']

for i, (m1, m2) in enumerate(metrics):
    ax[i].plot(cluster_scores['n_clusters'], cluster_scores[m1],label=m1)
    if m2: ax[i].plot(cluster_scores['n_clusters'], cluster_scores[m2],label=m2)
    ax[i].set_title(titles[i])
    ax[i].legend(), ax[i].grid(alpha=0.5)

plt.tight_layout()
plt.show()
```


Silhouette Score
0.2
0.1
0.0
−0.1
5
10
15
20
25
30
ss_agglom
ss_kmeans

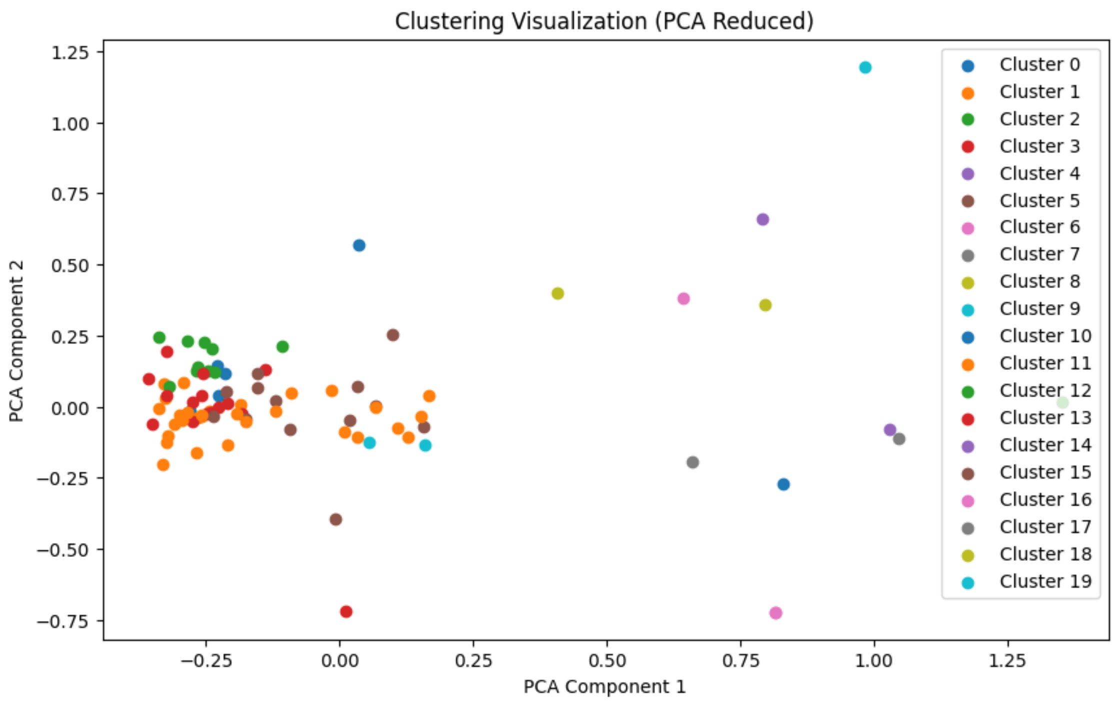

Davies-Bouldin Score
3.0
2.5
2.0
1.5
1.0
0.5
5
10
15
20
25
30
db_agglom
db_kmeans


SSD for K-Means
16000
14000
12000
10000
8000
6000
4000
5
10
15
20
25
30
ssd_kmeans

[63]:
```
# https://www.geeksforgeeks.org/
 ↪what-are-some-common-methods-for-determining-the-optimal-number-of-clusters/

# Choose the best num of clusters based on silhouette scores
best_n_agglom = cluster_scores.iloc[cluster_scores['ss_agglom'].
 ↪idxmax()]['n_clusters']
best_n_kmeans = cluster_scores.iloc[cluster_scores['ss_kmeans'].
 ↪idxmax()]['n_clusters']

print(f"Best n_clusters for Agglomerative Clustering: {best_n_agglom}")
print(f"Best n_clusters for K-Means: {best_n_kmeans}")
```

```
Best n_clusters for Agglomerative Clustering: 16.0
Best n_clusters for K-Means: 29.0
```

[64]:
```
# My own knowledge with help from docs https://scikit-learn.org/stable/

clusters = KMeans(n_clusters=int(best_n_kmeans)).fit(top_100_sc)

# Reduce dims
X_reduced = PCA(n_components=2).fit_transform(top_100_sc)

# Scatter plot
plt.figure(figsize=(10, 6))
for i in range(20):
    plt.scatter(X_reduced[clusters.labels_ == i, 0], X_reduced[clusters.labels_␣
 ↪== i, 1], label=f'Cluster {i}')

plt.legend()
plt.title("Clustering Visualization (PCA Reduced)")
plt.xlabel("PCA Component 1")
plt.ylabel("PCA Component 2")
plt.show()
```



Print the top 5 closest docs to each centroid

[65]:
```python
# My own knowledge
# Calc closest docs to each centroid
closest_idxs = closest_docs_to_centroid(top_100_sc, clusters.cluster_centers_,␣
 ↪clusters.labels_)

# Print
for i in range(int(best_n_kmeans)):
    print(f'Agglom cluster {i}: {np.sum(clusters.labels_ == i)}')
    for idx in closest_idxs[i]:
        print(f'Rating : {ratings[idx]} Review: {" ".join(clean_text[idx])}')
    print(50*'-')
```

```
Agglom cluster 0: 5
Rating : 2 Review: wanted know ring like 2 rings one ring beyond gorgeous love
Rating : 4 Review: nice small sized ring stick rings different looks
Rating : 3 Review: ring pretty enough metal ring substantial pushes easily
Rating : 5 Review: owned triple roll ring three rings slowly broke purchased far
none individual rings broken yet
Rating : 3 Review: ring pretty flexible turned good ordered ring finger size
made look better pink ring
--------------------------------------------------
Agglom cluster 1: 11
Rating : 4 Review: product came fast like amazon explained ring clearly written
```

```
war thick likebut guess like small rings ring
Rating : 5 Review: love ring shines beautiful looks real anyone like buy one
ring stones good size friends love ring could pass real diamond love ring worth
money better ring bought von silver cubic shine ring ring bought 3 stones worth
price anne marie
Rating : 3 Review: first skeptically ring price low nt wear lot gold purchased
ring nice gold necklaces wear wanted ring work ring nt look like 35 ring
Rating : 5 Review: item wonderful surprise quality much could ever hoped for
Rating : 5 Review: impressed quality item delivery fast would definitely buy
seller again
--------------------------------------------------
Agglom cluster 2: 10
Rating : 5 Review: ordered ring last december opened ring box arrived absolutely
took breath away gorgeous ring pictures nt justice size color stone suit taste
perfectly quality ring excellent quite variety silver rings ring one wear
regular basis taste rings toward boldunique side ring definitely fits bill
Rating : 5 Review: unfortunately small woman almost impossible purchase rings
without sized purchased ring would something wear vacation things could
otherwise harm expensive rings dont even need expensive rings ring stunning
delicate beautiful ring fits looks awesum
Rating : 5 Review: second engagement king got birthday love nice shiny must buy
ring anyone
Rating : 5 Review: checked around jewel stores prices could nt find rings could
even compare not huge ring definitely not microscopic either bought ring go
along solitary engagement ring makes perfect replacement plain wedding band
really glad decided go ring even though kind nervous ordering ring online
Rating : 5 Review: neck ring timely manner looks antique would recommend ring
garden lover
--------------------------------------------------
Agglom cluster 3: 14
Rating : 1 Review: ring beautiful ring first shipment ring scratches diamond
missing 2 diamonds shoulder ring returned ring replacement ring second ring even
worse clearly visible white scratch right middle black diamond not look great
time 1 diamond missing shoulder ring returned refund thanks clot amazon
Rating : 5 Review: love rings need pink rings hard find size 4 rings nt children
ring nice nice flower ring perfect pink ring me
Rating : 5 Review: ought ring thumb ring everyone notices comments pretty cool
looks spin want know got would recommend ring one also come
Rating : 5 Review: got not engagement ring much surprise two rings fit together
make 1 looked like one ring picture must say beautiful ring always catches
peoples attention
Rating : 5 Review: got ring promise ring girlfriend christmas loved definitely
great value
--------------------------------------------------
Agglom cluster 4: 1
Rating : 4 Review: sparkle pretty dainty must looking for
--------------------------------------------------
Agglom cluster 5: 12
```

```
Rating : 5 Review: recently lost ring much like one searching high low another
ring replace one lost came across ring ring absolutely brilliant ring look
shines sparkles like little diamonds love ring would recommend anyone wants ring
element adorable time
Rating : 3 Review: got ring birthday love not imagine woman not adoring ring
Rating : 5 Review: got ring birthday love not imagine woman not adoring ring
Rating : 4 Review: attractive high quality item young teenager small adult
Rating : 5 Review: addition wonderful sending ring ring beautiful daughter
thrilled ring thank dorothy
--------------------------------------------------
Agglom cluster 6: 2
Rating : 4 Review: husband wears size 10 ring tight new thumb ring guess nt
order things online cut ring
Rating : 4 Review: told ring comfortable wear quite surprised please see masonic
ring titanic
--------------------------------------------------
Agglom cluster 7: 1
Rating : 5 Review: think necklace well worth money delicate simple yet pretty
layed beautiful look
--------------------------------------------------
Agglom cluster 8: 1
Rating : 4 Review: love ring complaint metal soft bent point wear anymore still
wear necklace love occasion bluish onto figure took less two months almost
impossible wear
--------------------------------------------------
Agglom cluster 9: 1
Rating : 4 Review: solidbeautiful ring expecting color picture disappointed
barely pink first saw thought lavender still pretty buy design not color
--------------------------------------------------
Agglom cluster 10: 1
Rating : 5 Review: fiance looked many different rings fell love ring everything
wanted engagement ring wearing awhile perfect regrets buying ring
--------------------------------------------------
Agglom cluster 11: 19
Rating : 5 Review: bought ring old date beautiful ring fair price guy looking
save money afford engagement ring definitely get this
Rating : 3 Review: nice small sized ring stick rings different looks
Rating : 2 Review: serves purpose seemed image lot prettier sparkling turned
wear shirt since not compliment anything wear
Rating : 2 Review: wore toe ring one day poof stone gone sits pile broken
sterling silver rings things
Rating : 3 Review: bought ring valentine gift could nt give since hands holding
heart symbol would reverse whenever ring spine 4 ring alternating facing
direction cut ring kept
--------------------------------------------------
Agglom cluster 12: 1
Rating : 5 Review: wife loves ring great gift extremely cheap high quality
--------------------------------------------------
```

```
Agglom cluster 13: 1
Rating : 1 Review: huge waste money month part holds symbol necklace broke
seriously not wasting money low quality item
--------------------------------------------------
Agglom cluster 14: 1
Rating : 4 Review: nt even ordered ring read reviews notice picture 1 pictures
23 two different rings going order ring seems ring received andrd ring smaller
cross design
--------------------------------------------------
Agglom cluster 15: 1
Rating : 1 Review: hard ca nt wear material hard not easy wear nt like saved
money got
--------------------------------------------------
Agglom cluster 16: 1
Rating : 3 Review: love ring not ring would want wear everyday ring claws hold
stone sharp get stuck things scratched me
--------------------------------------------------
Agglom cluster 17: 1
Rating : 5 Review: not ring beautiful jeweller accommadating ring reach us time
appreciate care quick receiving ring thank
--------------------------------------------------
Agglom cluster 18: 1
Rating : 5 Review: ring awesome wear ring index finger fits great comfortable
ring
--------------------------------------------------
Agglom cluster 19: 2
Rating : 5 Review: see person appreciate beautiful really shiny feel pretty look
expensive much costthey also go color beautiful
Rating : 5 Review: stem great quality came promptly happy recommend undeservedly
--------------------------------------------------
Agglom cluster 20: 1
Rating : 3 Review: ring little small ring finger would better pink ring
--------------------------------------------------
Agglom cluster 21: 1
Rating : 5 Review: flute charm detailed high quality see keys flute fan would
adore item
--------------------------------------------------
Agglom cluster 22: 5
Rating : 2 Review: wanted not favorite piercing mine wear bioplast cut break
certain metals
Rating : 5 Review: still wearing pink ring trinity knotfinally pink ring not
turn finger green
Rating : 5 Review: ring perfect say spend thousands nt ring shines perfectly
love ring
Rating : 1 Review: ordered ring stated toe king description ring came quickly
not toe ring nt know anyone size 8 toe know anyone please direct seller wedding
band toe ring thumb ring sick one
Rating : 5 Review: love ring flowers go way around ring fits ring finger ever
```

```
find right nt buy engagement ring use engagement ring not getting debt marriage
--------------------------------------------------
Agglom cluster 23: 1
Rating : 3 Review: not ring beautiful jeweller accommadating ring reach us time
appreciate care quick receiving ring thank
--------------------------------------------------
Agglom cluster 24: 1
Rating : 5 Review: impressed quality would not hesitate purchase items seller
service also exceptional
--------------------------------------------------
Agglom cluster 25: 1
Rating : 4 Review: ring shipped accordingly love shipping pace ring beautiful
thing nt care line looks like ring may resized including specification included
ring met expectations especially beginning ring
--------------------------------------------------
Agglom cluster 26: 1
Rating : 5 Review: first skeptically ring price low nt wear lot gold purchased
ring nice gold necklaces wear wanted ring work ring nt look like 35 ring
--------------------------------------------------
Agglom cluster 27: 1
Rating : 5 Review: not completely sure buying jewellery internet type gift like
see handle item decide pressed time like pictures item decided take chance glad
impressed quality appearance item arrived price low compared quality wife
pleased jewellery item
--------------------------------------------------
Agglom cluster 28: 1
Rating : 4 Review: similar ring auction site flaw lies middle part ring complete
ring cuts flexibility joint ended taking mine silver smith remove back section
middle ring really helped flexibility
--------------------------------------------------
```

Print each cluster with more than 5 docs - this is what will be used for clustering

[66]:
```
# My own knowledge

# Want all the docs in clusters w n >=5
docs_in_clusters = [np.where(clusters.labels_==i) for i in␣
 ↪range(int(best_n_kmeans))]
count_docs_in_clusters = [len(docs_in_clusters[i][0]) for i in␣
 ↪range(int(best_n_kmeans))]

#Store index of each cluster n>=5
counts=[]
for i, count in enumerate(count_docs_in_clusters):
  if count>=5:
    counts.append(i)

cluster_text_list=[[] for _ in counts]
```

```
for j,i in enumerate(counts):
  print(f'Cluster {i}: {np.sum(clusters.labels_ == i)}')
  for idx in docs_in_clusters[i][0]: #idx is the index for the doc location in␣
↪top_100, not in clean_text which is what we want
    #so get the actual id and convert to string
    clean_text_ids = top_100_ids[idx]
    clean_text_ids = str(clean_text_ids)
    # Then get the index in clean_text - now can access the doc
    clean_text_idx = ids.index(clean_text_ids)

    # Store doc and print
    cluster_text_list[j].append(" ".join(clean_text[clean_text_idx]))
    print(f'Rating : {ratings[clean_text_idx]} Review: {" ".
↪join(clean_text[clean_text_idx])}')

  print(50*'-')
```

```
Cluster 0: 5
Rating : 5 Review: great gift loved one ever better seller seller deals
professional way security measures superb
Rating : 5 Review: product made great gift great memories love something always
helping gift heart always shows care
Rating : 5 Review: love ring fits right showed daughter ring loved well great
everyday wear price great
Rating : 4 Review: always love pillow free make great gifts great people life
quite collection hope continue build
Rating : 5 Review: mother loved great birthday gift look even better person go
great anything
--------------------------------------------------
Cluster 1: 11
Rating : 1 Review: product hollow not clearly specified item description not
expecting seemed poor quality returned item
Rating : 5 Review: impressed quality item delivery fast would definitely buy
seller again
Rating : 5 Review: flute charm detailed high quality see keys flute fan would
adore item
Rating : 1 Review: item not pictured funny poor quality seller not respond
contracted this
Rating : 1 Review: disappointed quality item fragile thin discoloured back charm
not returnable not ordering amazon jeweller future thank you
Rating : 1 Review: quality look not anticipated fliesi would not recommend item
Rating : 4 Review: attractive high quality item young teenager small adult
Rating : 1 Review: quality item not expectationsthe top scratched hinges not
line predrilled holes staining inconsistent saw item store would not purchased
Rating : 5 Review: pleased quality item definitely recommend addition friends
family
```

```
Rating : 2 Review: week one jewels fell wing metal already vanishing spend money
higher quality item
Rating : 1 Review: item misrepresented size quality horrible would return item
except family member coast guard sent total waste money
--------------------------------------------------
Cluster 2: 10
Rating : 5 Review: item perfect daily wear still elegant enough dress receive
many compliments wearing it
Rating : 2 Review: serves purpose seemed image lot prettier sparkling turned
wear shirt since not compliment anything wear
Rating : 5 Review: bought wear alice wonderland costume halloween wearing ever
since received actually gotten lot compliments
Rating : 4 Review: peasant classify best casual wear wear weekend nt not suited
work going events
Rating : 5 Review: nice wear want something casual wear comfortable
Rating : 5 Review: unique pleasure wear stones catch light style comfortable
wear
Rating : 4 Review: wear charm memory 3 friends passed away wish nt wear honor do
Rating : 5 Review: label pin perfect detail wear colors plan wear label wear
suit penis nice enough wear formal occasionsdear pride
Rating : 5 Review: days not wear blue one wear one really enjoy wearing
something celtic pretty
Rating : 5 Review: wanted class piece wear right hand work wearing old found end
wearing outside work class looking
--------------------------------------------------
Cluster 3: 14
Rating : 5 Review: arrived estimated date many previous orders seller always
time excellent condition
Rating : 5 Review: stem great quality came promptly happy recommend undeservedly
Rating : 5 Review: impressed quality would not hesitate purchase items seller
service also exceptional
Rating : 4 Review: item came quickly plenty time christmas huge hit person
received them
Rating : 5 Review: item wonderful surprise quality much could ever hoped for
Rating : 4 Review: told ring comfortable wear quite surprised please see masonic
ring titanic
Rating : 5 Review: wonderful shopping experience purchased item holiday presents
whole order came quickly wonderful condition
Rating : 5 Review: works perfectly rings wider rings narrow wide
Rating : 5 Review: always wanted claddaugh ringhis price greati love it
Rating : 5 Review: great way support local pro sports team without wearing
oversized jersey hat mess hair
Rating : 5 Review: good everyday wear dressing up
Rating : 3 Review: husband loves thing ring resided due way ring made
Rating : 5 Review: pleased purchase speedy shipping use
Rating : 5 Review: nice looks picture like
--------------------------------------------------
Cluster 5: 12
```

```
Rating : 4 Review: birthday gift 16 niece loves ring happy received it
Rating : 5 Review: wife loves ring great gift extremely cheap high quality
Rating : 5 Review: nice loves birth stone got christmas lifti also love also
great
Rating : 5 Review: got ring gift boyfriend love thing rings not position
correctly inches skin
Rating : 4 Review: picture rings shows blue sapphires disappointed see ring
received dark stones appear black ring definitely not advertised
Rating : 5 Review: got ring promise ring girlfriend christmas loved definitely
great value
Rating : 5 Review: ring good sparkle looks like ring cost six amount takes great
gift someone budget girlfriend loves it
Rating : 5 Review: bought ring husband loved received said would great ring
Rating : 2 Review: wore toe ring one day poof stone gone sits pile broken
sterling silver rings things
Rating : 2 Review: vice ring expensive ring one stone missing not worth
returning would pay postage ring cost
Rating : 5 Review: second engagement king got birthday love nice shiny must buy
ring anyone
Rating : 5 Review: bought gift friends birthday loved beautiful ring
--------------------------------------------------
Cluster 11: 19
Rating : 5 Review: purchased several items petya not products first rate packing
customer service excellent
Rating : 5 Review: product received nice came timely matter faster expected
order item again
Rating : 4 Review: received italian horn printing condition completely satisfied
receiving product timely manner
Rating : 3 Review: product arrived short period time perfect described perfectly
everything hoped
Rating : 2 Review: stem arrived extremely damaged several places not package
well send back disappointed quality
Rating : 4 Review: product arrived perfect condition shipping ridiculously slow
not order again
Rating : 4 Review: product came fast like amazon explained ring clearly written
war thick likebut guess like small rings ring
Rating : 1 Review: ring looks nothing like picture diamonds small not noticeable
sending back
Rating : 2 Review: stem shipped received within time limit given good quality
product t
Rating : 5 Review: earrings described transaction smooth easy product shipped
received time frame quoted pleased purchase
Rating : 4 Review: message positive looks pretty bought aunt present color nice
Rating : 4 Review: solidbeautiful ring expecting color picture disappointed
barely pink first saw thought lavender still pretty buy design not color
Rating : 1 Review: nt like product diamonds looked nothing like picture diamonds
flowed little bit
Rating : 5 Review: owned several claddagh rings years lost last one received one
```

```
gift recently truly love sturdy design crisp easily recommend one lovely ring
Rating : 5 Review: looked well picture thing could say little polished looks
like black stands looks nice
Rating : 5 Review: ring lovely advertised pink ring actually wear thumb ring
Rating : 5 Review: ring perfect say spend thousands nt ring shines perfectly
love ring
Rating : 3 Review: one beautiful rosary seen smoothness color beads translucent
looking almost looks like glass workmanship excellent details beautiful truly
beautiful piece own
Rating : 5 Review: look quite like photograph colourful know turtle seen
elsewhere quite high price beautiful
--------------------------------------------------
Cluster 22: 5
Rating : 5 Review: happy product received advertised timely manner selleramazon
kept updated shipmentdelivery status would recommend item seller
Rating : 5 Review: neck ring timely manner looks antique would recommend ring
garden lover
Rating : 2 Review: king way small looks like toy putting would not recommend
want nice 12 cart ring
Rating : 5 Review: ought ring thumb ring everyone notices comments pretty cool
looks spin want know got would recommend ring one also come
Rating : 3 Review: love ring not ring would want wear everyday ring claws hold
stone sharp get stuck things scratched me
--------------------------------------------------
```

4.2 Comments on results

Looking at the PCA cluster visualisation, the data itself is not too well clustered, with several lone points scattered on the graph. This explains the model's confusion, and why it assigned only one point to several different centroids. Generally, there appears to be four or five clusters with more than 5 docs, and these are the main clusters.

Every time this algorithm is run, it provides slightly different results. Therefore I chose to hard code the clusters I used for summarisation below. Additionally, because of the volume of clusters with only one point, I chose to only summarise the clusters with 5 or more docs.

5 Question 4

6 Summarisation

6.0.1 Extractive method = SBert

6.0.2 Abstractive method = Seq2Seq

[22]:
```
# Imports and Downloads
!pip install -U -q sentence-transformers
!pip install bert-extractive-summarizer
!pip install -U evaluate
```

```
!pip install setuptools==65.5.0
!pip install -U rouge_score
!pip install -U nltk
!pip install datasets
from transformers import TFAutoModelForSeq2SeqLM, DataCollatorForSeq2Seq,␣
↪AutoTokenizer
import evaluate
from sklearn.model_selection import train_test_split
import tensorflow as tf
from datasets import Dataset
from transformers import AdamWeightDecay
from transformers.keras_callbacks import KerasMetricCallback
from summarizer.sbert import SBertSummarizer
from rouge_score import rouge_scorer
from nltk.translate.meteor_score import meteor_score
from nltk.translate.bleu_score import sentence_bleu, SmoothingFunction

# Download necessary NLTK data
nltk.download('wordnet', quiet=True)
nltk.download('omw-1.4', quiet=True)
```

```
Requirement already satisfied: bert-extractive-summarizer in
/usr/local/lib/python3.11/dist-packages (0.10.1)
Requirement already satisfied: transformers in /usr/local/lib/python3.11/dist-
packages (from bert-extractive-summarizer) (4.48.3)
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.11/dist-
packages (from bert-extractive-summarizer) (1.6.1)
Requirement already satisfied: spacy in /usr/local/lib/python3.11/dist-packages
(from bert-extractive-summarizer) (3.7.5)
Requirement already satisfied: numpy>=1.19.5 in /usr/local/lib/python3.11/dist-
packages (from scikit-learn->bert-extractive-summarizer) (1.26.4)
Requirement already satisfied: scipy>=1.6.0 in /usr/local/lib/python3.11/dist-
packages (from scikit-learn->bert-extractive-summarizer) (1.14.1)
Requirement already satisfied: joblib>=1.2.0 in /usr/local/lib/python3.11/dist-
packages (from scikit-learn->bert-extractive-summarizer) (1.4.2)
Requirement already satisfied: threadpoolctl>=3.1.0 in
/usr/local/lib/python3.11/dist-packages (from scikit-learn->bert-extractive-
summarizer) (3.5.0)
Requirement already satisfied: spacy-legacy<3.1.0,>=3.0.11 in
/usr/local/lib/python3.11/dist-packages (from spacy->bert-extractive-summarizer)
(3.0.12)
Requirement already satisfied: spacy-loggers<2.0.0,>=1.0.0 in
/usr/local/lib/python3.11/dist-packages (from spacy->bert-extractive-summarizer)
(1.0.5)
Requirement already satisfied: murmurhash<1.1.0,>=0.28.0 in
/usr/local/lib/python3.11/dist-packages (from spacy->bert-extractive-summarizer)
(1.0.12)
```

```
Requirement already satisfied: cymem<2.1.0,>=2.0.2 in
/usr/local/lib/python3.11/dist-packages (from spacy->bert-extractive-summarizer)
(2.0.11)
Requirement already satisfied: preshed<3.1.0,>=3.0.2 in
/usr/local/lib/python3.11/dist-packages (from spacy->bert-extractive-summarizer)
(3.0.9)
Requirement already satisfied: thinc<8.3.0,>=8.2.2 in
/usr/local/lib/python3.11/dist-packages (from spacy->bert-extractive-summarizer)
(8.2.5)
Requirement already satisfied: wasabi<1.2.0,>=0.9.1 in
/usr/local/lib/python3.11/dist-packages (from spacy->bert-extractive-summarizer)
(1.1.3)
Requirement already satisfied: srsly<3.0.0,>=2.4.3 in
/usr/local/lib/python3.11/dist-packages (from spacy->bert-extractive-summarizer)
(2.5.1)
Requirement already satisfied: catalogue<2.1.0,>=2.0.6 in
/usr/local/lib/python3.11/dist-packages (from spacy->bert-extractive-summarizer)
(2.0.10)
Requirement already satisfied: weasel<0.5.0,>=0.1.0 in
/usr/local/lib/python3.11/dist-packages (from spacy->bert-extractive-summarizer)
(0.4.1)
Requirement already satisfied: typer<1.0.0,>=0.3.0 in
/usr/local/lib/python3.11/dist-packages (from spacy->bert-extractive-summarizer)
(0.15.2)
Requirement already satisfied: tqdm<5.0.0,>=4.38.0 in
/usr/local/lib/python3.11/dist-packages (from spacy->bert-extractive-summarizer)
(4.67.1)
Requirement already satisfied: requests<3.0.0,>=2.13.0 in
/usr/local/lib/python3.11/dist-packages (from spacy->bert-extractive-summarizer)
(2.32.3)
Requirement already satisfied: pydantic!=1.8,!=1.8.1,<3.0.0,>=1.7.4 in
/usr/local/lib/python3.11/dist-packages (from spacy->bert-extractive-summarizer)
(2.10.6)
Requirement already satisfied: jinja2 in /usr/local/lib/python3.11/dist-packages
(from spacy->bert-extractive-summarizer) (3.1.6)
Requirement already satisfied: setuptools in /usr/local/lib/python3.11/dist-
packages (from spacy->bert-extractive-summarizer) (65.5.0)
Requirement already satisfied: packaging>=20.0 in
/usr/local/lib/python3.11/dist-packages (from spacy->bert-extractive-summarizer)
(24.2)
Requirement already satisfied: langcodes<4.0.0,>=3.2.0 in
/usr/local/lib/python3.11/dist-packages (from spacy->bert-extractive-summarizer)
(3.5.0)
Requirement already satisfied: filelock in /usr/local/lib/python3.11/dist-
packages (from transformers->bert-extractive-summarizer) (3.17.0)
Requirement already satisfied: huggingface-hub<1.0,>=0.24.0 in
/usr/local/lib/python3.11/dist-packages (from transformers->bert-extractive-
summarizer) (0.28.1)
```

```
Requirement already satisfied: pyyaml>=5.1 in /usr/local/lib/python3.11/dist-
packages (from transformers->bert-extractive-summarizer) (6.0.2)
Requirement already satisfied: regex!=2019.12.17 in
/usr/local/lib/python3.11/dist-packages (from transformers->bert-extractive-
summarizer) (2024.11.6)
Requirement already satisfied: tokenizers<0.22,>=0.21 in
/usr/local/lib/python3.11/dist-packages (from transformers->bert-extractive-
summarizer) (0.21.0)
Requirement already satisfied: safetensors>=0.4.1 in
/usr/local/lib/python3.11/dist-packages (from transformers->bert-extractive-
summarizer) (0.5.3)
Requirement already satisfied: fsspec>=2023.5.0 in
/usr/local/lib/python3.11/dist-packages (from huggingface-
hub<1.0,>=0.24.0->transformers->bert-extractive-summarizer) (2024.10.0)
Requirement already satisfied: typing-extensions>=3.7.4.3 in
/usr/local/lib/python3.11/dist-packages (from huggingface-
hub<1.0,>=0.24.0->transformers->bert-extractive-summarizer) (4.12.2)
Requirement already satisfied: language-data>=1.2 in
/usr/local/lib/python3.11/dist-packages (from
langcodes<4.0.0,>=3.2.0->spacy->bert-extractive-summarizer) (1.3.0)
Requirement already satisfied: annotated-types>=0.6.0 in
/usr/local/lib/python3.11/dist-packages (from
pydantic!=1.8,!=1.8.1,<3.0.0,>=1.7.4->spacy->bert-extractive-summarizer) (0.7.0)
Requirement already satisfied: pydantic-core==2.27.2 in
/usr/local/lib/python3.11/dist-packages (from
pydantic!=1.8,!=1.8.1,<3.0.0,>=1.7.4->spacy->bert-extractive-summarizer)
(2.27.2)
Requirement already satisfied: charset-normalizer<4,>=2 in
/usr/local/lib/python3.11/dist-packages (from
requests<3.0.0,>=2.13.0->spacy->bert-extractive-summarizer) (3.4.1)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.11/dist-
packages (from requests<3.0.0,>=2.13.0->spacy->bert-extractive-summarizer)
(3.10)
Requirement already satisfied: urllib3<3,>=1.21.1 in
/usr/local/lib/python3.11/dist-packages (from
requests<3.0.0,>=2.13.0->spacy->bert-extractive-summarizer) (2.3.0)
Requirement already satisfied: certifi>=2017.4.17 in
/usr/local/lib/python3.11/dist-packages (from
requests<3.0.0,>=2.13.0->spacy->bert-extractive-summarizer) (2025.1.31)
Requirement already satisfied: blis<0.8.0,>=0.7.8 in
/usr/local/lib/python3.11/dist-packages (from thinc<8.3.0,>=8.2.2->spacy->bert-
extractive-summarizer) (0.7.11)
Requirement already satisfied: confection<1.0.0,>=0.0.1 in
/usr/local/lib/python3.11/dist-packages (from thinc<8.3.0,>=8.2.2->spacy->bert-
extractive-summarizer) (0.1.5)
Requirement already satisfied: click>=8.0.0 in /usr/local/lib/python3.11/dist-
packages (from typer<1.0.0,>=0.3.0->spacy->bert-extractive-summarizer) (8.1.8)
Requirement already satisfied: shellingham>=1.3.0 in
```

```
/usr/local/lib/python3.11/dist-packages (from typer<1.0.0,>=0.3.0->spacy->bert-
extractive-summarizer) (1.5.4)
Requirement already satisfied: rich>=10.11.0 in /usr/local/lib/python3.11/dist-
packages (from typer<1.0.0,>=0.3.0->spacy->bert-extractive-summarizer) (13.9.4)
Requirement already satisfied: cloudpathlib<1.0.0,>=0.7.0 in
/usr/local/lib/python3.11/dist-packages (from weasel<0.5.0,>=0.1.0->spacy->bert-
extractive-summarizer) (0.21.0)
Requirement already satisfied: smart-open<8.0.0,>=5.2.1 in
/usr/local/lib/python3.11/dist-packages (from weasel<0.5.0,>=0.1.0->spacy->bert-
extractive-summarizer) (7.1.0)
Requirement already satisfied: MarkupSafe>=2.0 in
/usr/local/lib/python3.11/dist-packages (from jinja2->spacy->bert-extractive-
summarizer) (3.0.2)
Requirement already satisfied: marisa-trie>=1.1.0 in
/usr/local/lib/python3.11/dist-packages (from language-
data>=1.2->langcodes<4.0.0,>=3.2.0->spacy->bert-extractive-summarizer) (1.2.1)
Requirement already satisfied: markdown-it-py>=2.2.0 in
/usr/local/lib/python3.11/dist-packages (from
rich>=10.11.0->typer<1.0.0,>=0.3.0->spacy->bert-extractive-summarizer) (3.0.0)
Requirement already satisfied: pygments<3.0.0,>=2.13.0 in
/usr/local/lib/python3.11/dist-packages (from
rich>=10.11.0->typer<1.0.0,>=0.3.0->spacy->bert-extractive-summarizer) (2.18.0)
Requirement already satisfied: wrapt in /usr/local/lib/python3.11/dist-packages
(from smart-open<8.0.0,>=5.2.1->weasel<0.5.0,>=0.1.0->spacy->bert-extractive-
summarizer) (1.17.2)
Requirement already satisfied: mdurl~=0.1 in /usr/local/lib/python3.11/dist-
packages (from markdown-it-
py>=2.2.0->rich>=10.11.0->typer<1.0.0,>=0.3.0->spacy->bert-extractive-
summarizer) (0.1.2)
Requirement already satisfied: evaluate in /usr/local/lib/python3.11/dist-
packages (0.4.3)
Requirement already satisfied: datasets>=2.0.0 in
/usr/local/lib/python3.11/dist-packages (from evaluate) (3.3.2)
Requirement already satisfied: numpy>=1.17 in /usr/local/lib/python3.11/dist-
packages (from evaluate) (1.26.4)
Requirement already satisfied: dill in /usr/local/lib/python3.11/dist-packages
(from evaluate) (0.3.8)
Requirement already satisfied: pandas in /usr/local/lib/python3.11/dist-packages
(from evaluate) (2.2.2)
Requirement already satisfied: requests>=2.19.0 in
/usr/local/lib/python3.11/dist-packages (from evaluate) (2.32.3)
Requirement already satisfied: tqdm>=4.62.1 in /usr/local/lib/python3.11/dist-
packages (from evaluate) (4.67.1)
Requirement already satisfied: xxhash in /usr/local/lib/python3.11/dist-packages
(from evaluate) (3.5.0)
Requirement already satisfied: multiprocess in /usr/local/lib/python3.11/dist-
packages (from evaluate) (0.70.16)
Requirement already satisfied: fsspec>=2021.05.0 in
```

```
/usr/local/lib/python3.11/dist-packages (from fsspec[http]>=2021.05.0->evaluate)
(2024.10.0)
Requirement already satisfied: huggingface-hub>=0.7.0 in
/usr/local/lib/python3.11/dist-packages (from evaluate) (0.28.1)
Requirement already satisfied: packaging in /usr/local/lib/python3.11/dist-
packages (from evaluate) (24.2)
Requirement already satisfied: filelock in /usr/local/lib/python3.11/dist-
packages (from datasets>=2.0.0->evaluate) (3.17.0)
Requirement already satisfied: pyarrow>=15.0.0 in
/usr/local/lib/python3.11/dist-packages (from datasets>=2.0.0->evaluate)
(18.1.0)
Requirement already satisfied: aiohttp in /usr/local/lib/python3.11/dist-
packages (from datasets>=2.0.0->evaluate) (3.11.13)
Requirement already satisfied: pyyaml>=5.1 in /usr/local/lib/python3.11/dist-
packages (from datasets>=2.0.0->evaluate) (6.0.2)
Requirement already satisfied: typing-extensions>=3.7.4.3 in
/usr/local/lib/python3.11/dist-packages (from huggingface-hub>=0.7.0->evaluate)
(4.12.2)
Requirement already satisfied: charset-normalizer<4,>=2 in
/usr/local/lib/python3.11/dist-packages (from requests>=2.19.0->evaluate)
(3.4.1)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.11/dist-
packages (from requests>=2.19.0->evaluate) (3.10)
Requirement already satisfied: urllib3<3,>=1.21.1 in
/usr/local/lib/python3.11/dist-packages (from requests>=2.19.0->evaluate)
(2.3.0)
Requirement already satisfied: certifi>=2017.4.17 in
/usr/local/lib/python3.11/dist-packages (from requests>=2.19.0->evaluate)
(2025.1.31)
Requirement already satisfied: python-dateutil>=2.8.2 in
/usr/local/lib/python3.11/dist-packages (from pandas->evaluate) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.11/dist-
packages (from pandas->evaluate) (2025.1)
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.11/dist-
packages (from pandas->evaluate) (2025.1)
Requirement already satisfied: aiohappyeyeballs>=2.3.0 in
/usr/local/lib/python3.11/dist-packages (from
aiohttp->datasets>=2.0.0->evaluate) (2.5.0)
Requirement already satisfied: aiosignal>=1.1.2 in
/usr/local/lib/python3.11/dist-packages (from
aiohttp->datasets>=2.0.0->evaluate) (1.3.2)
Requirement already satisfied: attrs>=17.3.0 in /usr/local/lib/python3.11/dist-
packages (from aiohttp->datasets>=2.0.0->evaluate) (25.1.0)
Requirement already satisfied: frozenlist>=1.1.1 in
/usr/local/lib/python3.11/dist-packages (from
aiohttp->datasets>=2.0.0->evaluate) (1.5.0)
Requirement already satisfied: multidict<7.0,>=4.5 in
/usr/local/lib/python3.11/dist-packages (from
aiohttp->datasets>=2.0.0->evaluate) (6.1.0)
Requirement already satisfied: propcache>=0.2.0 in
/usr/local/lib/python3.11/dist-packages (from
aiohttp->datasets>=2.0.0->evaluate) (0.3.0)
Requirement already satisfied: yarl<2.0,>=1.17.0 in
/usr/local/lib/python3.11/dist-packages (from
aiohttp->datasets>=2.0.0->evaluate) (1.18.3)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.11/dist-
packages (from python-dateutil>=2.8.2->pandas->evaluate) (1.17.0)
Requirement already satisfied: setuptools==65.5.0 in
/usr/local/lib/python3.11/dist-packages (65.5.0)
Requirement already satisfied: rouge_score in /usr/local/lib/python3.11/dist-
packages (0.1.2)
Requirement already satisfied: absl-py in /usr/local/lib/python3.11/dist-
packages (from rouge_score) (1.4.0)
Requirement already satisfied: nltk in /usr/local/lib/python3.11/dist-packages
(from rouge_score) (3.9.1)
Requirement already satisfied: numpy in /usr/local/lib/python3.11/dist-packages
(from rouge_score) (1.26.4)
Requirement already satisfied: six>=1.14.0 in /usr/local/lib/python3.11/dist-
packages (from rouge_score) (1.17.0)
Requirement already satisfied: click in /usr/local/lib/python3.11/dist-packages
(from nltk->rouge_score) (8.1.8)
Requirement already satisfied: joblib in /usr/local/lib/python3.11/dist-packages
(from nltk->rouge_score) (1.4.2)
Requirement already satisfied: regex>=2021.8.3 in
/usr/local/lib/python3.11/dist-packages (from nltk->rouge_score) (2024.11.6)
Requirement already satisfied: tqdm in /usr/local/lib/python3.11/dist-packages
(from nltk->rouge_score) (4.67.1)
Requirement already satisfied: nltk in /usr/local/lib/python3.11/dist-packages
(3.9.1)
Requirement already satisfied: click in /usr/local/lib/python3.11/dist-packages
(from nltk) (8.1.8)
Requirement already satisfied: joblib in /usr/local/lib/python3.11/dist-packages
(from nltk) (1.4.2)
Requirement already satisfied: regex>=2021.8.3 in
/usr/local/lib/python3.11/dist-packages (from nltk) (2024.11.6)
Requirement already satisfied: tqdm in /usr/local/lib/python3.11/dist-packages
(from nltk) (4.67.1)
Requirement already satisfied: datasets in /usr/local/lib/python3.11/dist-
packages (3.3.2)
Requirement already satisfied: filelock in /usr/local/lib/python3.11/dist-
packages (from datasets) (3.17.0)
Requirement already satisfied: numpy>=1.17 in /usr/local/lib/python3.11/dist-
packages (from datasets) (1.26.4)
Requirement already satisfied: pyarrow>=15.0.0 in
/usr/local/lib/python3.11/dist-packages (from datasets) (18.1.0)
Requirement already satisfied: dill<0.3.9,>=0.3.0 in
```

```
/usr/local/lib/python3.11/dist-packages (from datasets) (0.3.8)
Requirement already satisfied: pandas in /usr/local/lib/python3.11/dist-packages
(from datasets) (2.2.2)
Requirement already satisfied: requests>=2.32.2 in
/usr/local/lib/python3.11/dist-packages (from datasets) (2.32.3)
Requirement already satisfied: tqdm>=4.66.3 in /usr/local/lib/python3.11/dist-
packages (from datasets) (4.67.1)
Requirement already satisfied: xxhash in /usr/local/lib/python3.11/dist-packages
(from datasets) (3.5.0)
Requirement already satisfied: multiprocess<0.70.17 in
/usr/local/lib/python3.11/dist-packages (from datasets) (0.70.16)
Requirement already satisfied: fsspec<=2024.12.0,>=2023.1.0 in
/usr/local/lib/python3.11/dist-packages (from
fsspec[http]<=2024.12.0,>=2023.1.0->datasets) (2024.10.0)
Requirement already satisfied: aiohttp in /usr/local/lib/python3.11/dist-
packages (from datasets) (3.11.13)
Requirement already satisfied: huggingface-hub>=0.24.0 in
/usr/local/lib/python3.11/dist-packages (from datasets) (0.28.1)
Requirement already satisfied: packaging in /usr/local/lib/python3.11/dist-
packages (from datasets) (24.2)
Requirement already satisfied: pyyaml>=5.1 in /usr/local/lib/python3.11/dist-
packages (from datasets) (6.0.2)
Requirement already satisfied: aiohappyeyeballs>=2.3.0 in
/usr/local/lib/python3.11/dist-packages (from aiohttp->datasets) (2.5.0)
Requirement already satisfied: aiosignal>=1.1.2 in
/usr/local/lib/python3.11/dist-packages (from aiohttp->datasets) (1.3.2)
Requirement already satisfied: attrs>=17.3.0 in /usr/local/lib/python3.11/dist-
packages (from aiohttp->datasets) (25.1.0)
Requirement already satisfied: frozenlist>=1.1.1 in
/usr/local/lib/python3.11/dist-packages (from aiohttp->datasets) (1.5.0)
Requirement already satisfied: multidict<7.0,>=4.5 in
/usr/local/lib/python3.11/dist-packages (from aiohttp->datasets) (6.1.0)
Requirement already satisfied: propcache>=0.2.0 in
/usr/local/lib/python3.11/dist-packages (from aiohttp->datasets) (0.3.0)
Requirement already satisfied: yarl<2.0,>=1.17.0 in
/usr/local/lib/python3.11/dist-packages (from aiohttp->datasets) (1.18.3)
Requirement already satisfied: typing-extensions>=3.7.4.3 in
/usr/local/lib/python3.11/dist-packages (from huggingface-hub>=0.24.0->datasets)
(4.12.2)
Requirement already satisfied: charset-normalizer<4,>=2 in
/usr/local/lib/python3.11/dist-packages (from requests>=2.32.2->datasets)
(3.4.1)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.11/dist-
packages (from requests>=2.32.2->datasets) (3.10)
Requirement already satisfied: urllib3<3,>=1.21.1 in
/usr/local/lib/python3.11/dist-packages (from requests>=2.32.2->datasets)
(2.3.0)
Requirement already satisfied: certifi>=2017.4.17 in
```

```
/usr/local/lib/python3.11/dist-packages (from requests>=2.32.2->datasets)
(2025.1.31)
Requirement already satisfied: python-dateutil>=2.8.2 in
/usr/local/lib/python3.11/dist-packages (from pandas->datasets) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.11/dist-
packages (from pandas->datasets) (2025.1)
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.11/dist-
packages (from pandas->datasets) (2025.1)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.11/dist-
packages (from python-dateutil>=2.8.2->pandas->datasets) (1.17.0)
```

[22]:
```
True
```

[23]:
```
# Hard coded clusters, as generated above
```

```
cluster_text_list = [['arrived estimated date many previous orders seller␣
 ↪always time excellent condition', 'happy product received advertised timely␣
 ↪manner selleramazon kept updated shipmentdelivery status would recommend␣
 ↪item seller', 'stem great quality came promptly happy recommend␣
 ↪undeservedly', 'purchased several items petya not products first rate␣
 ↪packing customer service excellent', 'product arrived short period time␣
 ↪perfect described perfectly everything hoped', 'product received nice came␣
 ↪timely matter faster expected order item again', 'neck ring timely manner␣
 ↪looks antique would recommend ring garden lover', 'received italian horn␣
 ↪printing condition completely satisfied receiving product timely manner',␣
 ↪'product arrived perfect condition shipping ridiculously slow not order␣
 ↪again', 'product came fast like amazon explained ring clearly written war␣
 ↪thick likebut guess like small rings ring', 'earrings described transaction␣
 ↪smooth easy product shipped received time frame quoted pleased purchase',␣
 ↪'stem shipped received within time limit given good quality product t',␣
```

```
# Extractive ref sums generated by Claude 13/03/2025
extractive_ref_sums = [
    'Product received nice came timely matter faster expected order item again.␣
 ↪Earrings described transaction smooth easy product shipped received time␣
 ↪frame quoted pleased purchase. Wonderful shopping experience purchased item␣
 ↪holiday presents whole order came quickly wonderful condition.',
    'Ought ring thumb ring everyone notices comments pretty cool looks spin␣
 ↪want know got would recommend ring one also come. Owned several claddagh␣
 ↪rings years lost last one received one gift recently truly love sturdy␣
 ↪design crisp easily recommend one lovely ring. Ring absolutely stunning␣
 ↪beautiful would recommend amethyst ring anyone market reasonably prices␣
 ↪amethyst ring.',
    'Item perfect daily wear still elegant enough dress receive many␣
 ↪compliments wearing it. Nice wear want something casual wear comfortable.␣
 ↪Unique pleasure wear stones catch light style comfortable wear.',
    'Impressed quality would not hesitate purchase items seller service also␣
 ↪exceptional. Flute charm detailed high quality see keys flute fan would␣
 ↪adore item. Item wonderful surprise quality much could ever hoped for.',
    'Great gift loved one ever better seller seller deals professional way␣
 ↪security measures superb. Ring good sparkle looks like ring cost six amount␣
 ↪takes great gift someone budget girlfriend loves it. Love birthstone wanted␣
 ↪piece jewel symbolized simple purity blue opal ring gift birthday year␣
 ↪definitely great gift welcomed addition collection.'
]


# abstractive ref sums generated by Claude 13/03/2025
abstractive_ref_summaries = [
    'Products arrive on time and in excellent condition, with customers␣
 ↪generally satisfied with shipping speed and product quality. Most reviewers␣
 ↪express high satisfaction with the rings, earrings and other jewelry items,␣
 ↪highlighting their appearance and value. While most customers are pleased␣
 ↪with their purchases, a small minority report quality issues such as missing␣
 ↪stones or differences from product images. Several customers specifically␣
 ↪praise the timely delivery and recommend both the products and sellers,␣
 ↪appreciating the customer service and transaction experience.',
    'Customers express strong satisfaction with their ring purchases, often␣
 ↪buying them as gifts or for personal use. Reviewers consistently highlight␣
 ↪the visual appeal of the rings, with multiple customers mentioning receiving␣
 ↪compliments from others. The claddagh rings are particularly appreciated for␣
 ↪their sturdy design, while amethyst rings are noted for being reasonably␣
 ↪priced and stunning. Overall, the body jewelry and rings in this cluster␣
 ↪receive enthusiastic recommendations from buyers.',
    'Reviews focus primarily on the wearability of jewelry items, with mixed␣
 ↪experiences. Some customers find their purchases comfortable and suitable␣
 ↪for daily wear, even receiving compliments. Others express disappointment␣
 ↪with comfort issues, mentioning sharp stone settings that get caught on␣
 ↪clothing or scratch skin. Several reviewers categorize their items as casual␣
 ↪everyday pieces versus those appropriate for dressing up. Celtic-themed␣
 ↪jewelry receives positive mentions for its uniqueness, while some items are␣
 ↪criticized for poor quality materials that affect wearability.',
    'Reviews in this cluster center on product quality with sharply divided␣
 ↪opinions. Satisfied customers praise the high quality, detailed␣
 ↪craftsmanship, and fast delivery of their purchases, often expressing␣
 ↪willingness to buy from the same sellers again. Dissatisfied customers␣
 ↪describe items as hollow, fragile, thin, or damaged upon arrival. Several␣
 ↪reviewers mention discrepancies between product images and the actual items␣
 ↪received. Poor packaging, discoloration, and missing jewels are common␣
 ↪complaints, with some customers expressing regret over their purchases and␣
 ↪reluctance to order similar items in the future.',
    'Reviews predominantly discuss jewelry as gifts, with overwhelmingly␣
 ↪positive feedback. Customers frequently purchase rings as birthday,␣
 ↪Christmas, or promise gifts for loved ones, reporting that recipients love␣
 ↪the items. Many reviewers highlight the good value, noting that the rings␣
 ↪look more expensive than their actual cost. Birthstone jewelry is␣
 ↪specifically mentioned as meaningful gifts with symbolic value. Several␣
 ↪customers mention that the items look even better in person than in pictures␣
 ↪and pair well with various outfits. The cluster reflects high satisfaction␣
 ↪with jewelry items as thoughtful, well-received gifts.'
]
```

[24]:
```
# https://github.com/gcosma/COP509/blob/main/LabSolutions/Lab_Exercise_(No_6).
 ↪ipynb

def evaluate_summarisation(summaries, extractive_ref_sums):
  '''Returns the average rouge1, rouge2, rougeL, meteor and bleu scores'''

  scorer = rouge_scorer.RougeScorer(['rouge1', 'rouge2', 'rougeL'],␣
 ↪use_stemmer=True)

  rouge1, rouge2, rougeL, meteor, bleu=[],[],[],[],[]
  for i, (sum, ref) in enumerate(zip(summaries, extractive_ref_sums)):

    scores = scorer.score(sum, ref)
    rouge1.append(scores['rouge1'].fmeasure)
    rouge2.append(scores['rouge2'].fmeasure)
    rougeL.append(scores['rougeL'].fmeasure)

    ref_tokens = word_tokenize(ref)
    sum_tokens = word_tokenize(sum)
    meteor.append(meteor_score([ref_tokens], sum_tokens))
    bleu.append(sentence_bleu([ref], sum,␣
↪smoothing_function=SmoothingFunction().method1))

  return np.mean(rouge1), np.mean(rouge2), np.mean(rougeL), np.mean(meteor), np.
↪mean(bleu)

def plot_eval_graph(scores_dict):
    '''Plot bar graph of all scores'''
    # Extract scores
    metrics = ['ROUGE-1', 'ROUGE-2', 'ROUGE-L', 'METEOR', 'BLEU']
    extractive_scores = scores_dict.get('Extractive', (0, 0, 0, 0, 0))
    abstractive_scores = scores_dict.get('Abstractive', (0, 0, 0, 0, 0))
    abstractive_pruned = scores_dict.get('Abstractive_pruned', (0, 0, 0, 0, 0))

    # Format widths
    x = np.arange(len(metrics))
    width = 0.25  # Adjusted for proper spacing

    fig, ax = plt.subplots(figsize=(10, 6))

    # Plot bars
    bars1 = ax.bar(x - width, extractive_scores, width, label='Extractive',␣
↪color='pink')
    bars2 = ax.bar(x, abstractive_scores, width, label='Abstractive',␣
↪color='purple')
    bars3 = ax.bar(x + width, abstractive_pruned, width,␣
↪label='Abstractive_pruned', color='skyblue')

    # Aesthetics
    ax.set_xlabel('Evaluation Metrics')
    ax.set_ylabel('Scores')
    ax.set_title('Extractive vs. Abstractive using Metrics (Avg across all␣
↪clusters)')
    ax.set_xticks(x)
    ax.set_xticklabels(metrics)
    ax.legend()
    ax.grid(alpha=0.5)

    # Add value labels on bars
    for bars in [bars1, bars2, bars3]:
        for bar in bars:
            height = bar.get_height()
            ax.annotate(f'{height:.2f}',
                        xy=(bar.get_x() + bar.get_width() / 2, height),  #␣
↪Center text above bar
```

```
                    xytext=(0, 3),  # Offset text slightly above the bar
                    textcoords='offset points',
                    ha='center', va='bottom', fontsize=10)

    plt.show()
```

[25]:
```
# Adapted from https://github.com/gcosma/COP509/blob/main/Tutorials/
 ↪Tutorial7Summarization_with_user_pasted_data.ipynb

model = SBertSummarizer('paraphrase-MiniLM-L6-v2')

# Summarize the text
summaries=[]
for clus in cluster_text_list:
  clus = '. '.join(clus) +'.' # Add full stops to aid model
  summaries.append(model(clus, num_sentences=3))

# Print the summaries
for i,summary in enumerate(summaries):
  print(f"Cluster summary {i}: " + summary + "\n")

# Get eval metrics (plotted later)
e_rouge1, e_rouge2, e_rougeL, e_meteor, e_bleu =␣
 ↪evaluate_summarisation(summaries, extractive_ref_sums)
```

```
modules.json:   0%|          | 0.00/229 [00:00<?, ?B/s]

config_sentence_transformers.json:   0%|          | 0.00/122 [00:00<?, ?B/s]

README.md:   0%|          | 0.00/3.51k [00:00<?, ?B/s]

sentence_bert_config.json:   0%|          | 0.00/53.0 [00:00<?, ?B/s]

config.json:   0%|          | 0.00/629 [00:00<?, ?B/s]

model.safetensors:   0%|          | 0.00/90.9M [00:00<?, ?B/s]

tokenizer_config.json:   0%|          | 0.00/314 [00:00<?, ?B/s]

vocab.txt:   0%|          | 0.00/232k [00:00<?, ?B/s]

tokenizer.json:   0%|          | 0.00/466k [00:00<?, ?B/s]

special_tokens_map.json:   0%|          | 0.00/112 [00:00<?, ?B/s]

config.json:   0%|          | 0.00/190 [00:00<?, ?B/s]

Cluster summary 0: arrived estimated date many previous orders seller always
time excellent condition. product received nice came timely matter faster
expected order item again. product came fast like amazon explained ring clearly
written war thick likebut guess like small rings ring.

Cluster summary 1: got 3 belly rings 3 tongue rings 3 lip rings fig car stuff
```

```
got cool love pink peace sign tongue ring complaint got 2 yellow lip rings not
much yellow overall happy. got ring gift boyfriend love thing rings not position
correctly inches skin. ring absolutely stunning beautiful would recommend
amethyst ring anyone market reasonably prices amethyst ring.

Cluster summary 2: told ring comfortable wear quite surprised please see masonic
ring titanic. item perfect daily wear still elegant enough dress receive many
compliments wearing it. days not wear blue one wear one really enjoy wearing
something celtic pretty.

Cluster summary 3: impressed quality would not hesitate purchase items seller
service also exceptional. item not pictured funny poor quality seller not
respond contracted this. item misrepresented size quality horrible would return
item except family member coast guard sent total waste money.

Cluster summary 4: great gift loved one ever better seller seller deals
professional way security measures superb. wife loves ring great gift extremely
cheap high quality. bought gift friends birthday loved beautiful ring.
```

6.1 Abstractive Summarisation

[26]:
```python
# Adapted from https://github.com/gcosma/COP509/blob/main/Tutorials/
↪Tutorial8Summarization.ipynb

def preprocess_function(examples):
    inputs = [prefix + doc for doc in examples["document"]]
    model_inputs = tokenizer(inputs, max_length=max_input_length,␣
↪truncation=True)

    # Setup the tokenizer for targets
    with tokenizer.as_target_tokenizer():
        labels = tokenizer(examples["summary"], max_length=max_target_length,␣
↪truncation=True)

    model_inputs["labels"] = labels["input_ids"]
    return model_inputs

# Specify the model checkpoint
model_checkpoint = "t5-small"

# Initialize the tokenizer
tokenizer = AutoTokenizer.from_pretrained(model_checkpoint)
model = TFAutoModelForSeq2SeqLM.from_pretrained(model_checkpoint)

# Load the ROUGE metric
rouge = evaluate.load('rouge')
```

```
# Train test split data
inputs = ['. '.join(doc) + '.' for doc in cluster_text_list]
data = pd.DataFrame({'document': inputs, 'summary': abstractive_ref_summaries,␣
  ↪'id':[27, 6, 12, 15, 13]})
train_as, test_as = train_test_split(data, test_size=0.2, random_state=42)

# Convert to hugging face transformers dataset
train_dataset = Dataset.from_pandas(train_as)
test_dataset = Dataset.from_pandas(test_as)

# Process the datasets
prefix = 'summarize'
max_input_length = 1024
max_target_length = 128

train_dataset = train_dataset.map(preprocess_function, batched=True)
test_dataset = test_dataset.map(preprocess_function, batched=True)
```

```
tokenizer_config.json:   0%|          | 0.00/2.32k [00:00<?, ?B/s]

spiece.model:   0%|          | 0.00/792k [00:00<?, ?B/s]

tokenizer.json:   0%|          | 0.00/1.39M [00:00<?, ?B/s]

config.json:   0%|          | 0.00/1.21k [00:00<?, ?B/s]

model.safetensors:   0%|          | 0.00/242M [00:00<?, ?B/s]

All PyTorch model weights were used when initializing
TFT5ForConditionalGeneration.

All the weights of TFT5ForConditionalGeneration were initialized from the
PyTorch model.
If your task is similar to the task the model of the checkpoint was trained on,
you can already use TFT5ForConditionalGeneration for predictions without further
training.

Downloading builder script:   0%|          | 0.00/6.27k [00:00<?, ?B/s]

Map:   0%|          | 0/4 [00:00<?, ? examples/s]

Map:   0%|          | 0/1 [00:00<?, ? examples/s]
```

[27]:
```
# Adapted from https://github.com/gcosma/COP509/blob/main/Tutorials/
  ↪Tutorial8Summarization.ipynb

batch_size = 1
learning_rate = 2e-5
weight_decay = 0.01
num_train_epochs = 1
```

```
data_collator = DataCollatorForSeq2Seq(tokenizer, model=model,␣
↪return_tensors="np")

generation_data_collator = DataCollatorForSeq2Seq(tokenizer, model=model,␣
↪return_tensors="np", pad_to_multiple_of=128)

train_dataset = model.prepare_tf_dataset(
    train_dataset,
    batch_size=batch_size,
    shuffle=True,
    collate_fn=data_collator,
)

validation_dataset = model.prepare_tf_dataset(
    test_dataset,
    batch_size=batch_size,
    shuffle=False,
    collate_fn=data_collator,
)
```

[28]:

```
# Adapted from https://github.com/gcosma/COP509/blob/main/Tutorials/
↪Tutorial8Summarization.ipynb

optimizer = AdamWeightDecay(learning_rate=learning_rate,␣
↪weight_decay_rate=weight_decay)
model.compile(optimizer=optimizer)

# Define the metric function
rouge = evaluate.load('rouge')

def metric_fn(eval_pred):
    predictions, labels = eval_pred
    decoded_preds = tokenizer.batch_decode(predictions,␣
↪skip_special_tokens=True)
    decoded_labels = tokenizer.batch_decode(labels, skip_special_tokens=True)
    return rouge.compute(predictions=decoded_preds, references=decoded_labels)

# Set up metric callback
metric_callback = KerasMetricCallback(
    metric_fn=metric_fn,
    eval_dataset=validation_dataset,
    predict_with_generate=True,
    use_xla_generation=True
)

# Train the model with just the metric callback
```

```
model.fit(
    train_dataset,
    validation_data=validation_dataset,
    epochs=1,
    callbacks=[metric_callback]
)
```

```
4/4 [==============================] - 92s 11s/step - loss: 5.1762 - val_loss:
4.1216 - rouge1: 0.0476 - rouge2: 0.0000e+00 - rougeL: 0.0476 - rougeLsum:
0.0476
```

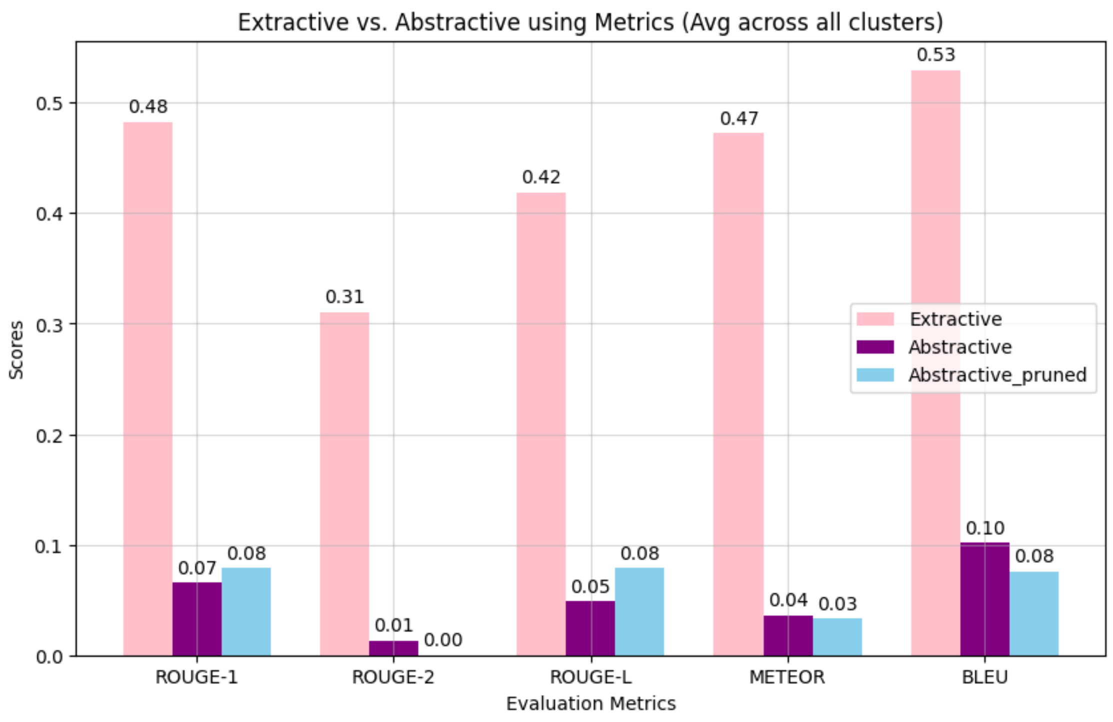
```
[28]: <tf_keras.src.callbacks.History at 0x7a0ce75e8050>
```

```
[29]: # Adapted from https://github.com/gcosma/COP509/blob/main/Tutorials/
       ↪Tutorial8Summarization.ipynb

      # Generate summaries
      abs_sums = []
      for i, docs in enumerate(cluster_text_list):
        documents = ['summarize' + document for document in docs]

        tokenized = tokenizer(' '.join(documents), return_tensors='np')
        out = model.generate(**tokenized, max_length=128)

        with tokenizer.as_target_tokenizer():
          abs_sums.append(tokenizer.decode(out[0]))
          print(f'Summary {i}: {tokenizer.decode(out[0])}')

      # Evaluate
      a_rouge1, a_rouge2, a_rougeL, a_meteor, a_bleu =␣
       ↪evaluate_summarisation(abs_sums, abstractive_ref_summaries)
      a_rouge1_pruned, a_rouge2_pruned, a_rougeL_pruned, a_meteor_pruned,␣
       ↪a_bleu_pruned = evaluate_summarisation(abs_sums, abstractive_ref_summaries[1:
       ↪2])
```

```
Summary 0: <pad> ring ring ring ring ring ring ring ring ring ring ring ring
ring ring ring ring ring ring ring ring ring ring ring ring ring ring ring ring
ring ring ring ring ring ring ring ring ring ring ring ring ring ring ring ring
ring ring ring ring ring ring ring ring ring ring ring ring </s>
Summary 1: <pad> ring one also come summarized several claddagh rings years lost
last one received one gift recently truly love sturdy design crisp easily
recommend one lovely ring summarized second engagement king got birthday love
nice shiny must buy ring anyone market reasonably prices amethyst ring anyone
market reasonably prices amethyst ring.</s>
Summary 2: <pad> ring comfortable wear nt like saved money nt wear honor do nt
wear honor do nt wear honor do nt wear honor do nt wear nt like saved money nt
wear nt like saved money nt wear nt like saved money nt wear nt like saved money
```

```
nt wear nt like saved money nt wear nt like saved money nt wear nt like saved
money nt wear nt like saved</s>
Summary 3: <pad><extra_id_0> a<extra_id_1> a<extra_id_2> item<extra_id_3>
a<extra_id_4> a<extra_id_5> a<extra_id_6> a<extra_id_7> a<extra_id_8>
a<extra_id_9> a<extra_id_10> a<extra_id_11> a<extra_id_12> a<extra_id_13>
a<extra_id_14> a<extra_id_15> a<extra_id_16> a<extra_id_17> a<extra_id_18> ad ad
ad ad ad ad ad ad ad ad ad ad ad ad ad ad ad ad ad a</s>
Summary 4: <pad> ring ring ring ring ring ring ring ring ring ring ring ring
ring ring ring ring ring ring ring ring ring ring ring ring ring ring ring ring
ring ring ring ring ring ring ring ring ring ring ring ring ring ring ring ring
ring ring ring ring ring ring ring ring ring ring ring ring </s>
```

[30]:
```python
plot_eval_graph({'Extractive': (e_rouge1, e_rouge2, e_rougeL, e_meteor, e_bleu),
                 'Abstractive': (a_rouge1, a_rouge2, a_rougeL, a_meteor,␣
  ↪a_bleu),
                  'Abstractive_pruned': (a_rouge1_pruned, a_rouge2_pruned,␣
  ↪a_rougeL_pruned, a_meteor_pruned, a_bleu_pruned)})
```



6.1.1 Summarisation comments

The extractive summarisation techniques were significantly better. The abstractive model was not trained on enough data. Some of the summaries were repetitions of 'ring', so the model was not trained enough to generalise well. Only two of the abstracted summaries were readable, so I plotted the scores for just those in an `abstractive_pruned` set. The scores only barely increased, and were in some cases worse (ROUGE-2, METEOR)

6.2 PyTerrier attempt at NIR

[ ]:
```
# !pip install transformers
# !pip install torch
# !pip install python-terrier
# !pip install git+https://github.com/terrierteam/pyterrier_colbert.git

# import pyterrier as pt
# if not pt.started():
#     pt.init()

# import os
# from transformers import AutoTokenizer, AutoModel

# from pyterrier_colbert.indexing import ColBERTIndexer
# from pyterrier_colbert.ranking import ColBERTFactory
# from collections import Counter
```

[ ]:
```
# checkpoint="http://www.dcs.gla.ac.uk/~craigm/colbert.dnn.zip"

# def prepare_dataset(ids, clean_text, query_list, query_relevant):
#   queries=pt.new.queries(query_list, range(1, len(query_list)+1))

#   docs=[]
#   for idx in range(0,len(clean_text)):
#     if len(clean_text[idx]) >1:
#       docs.append({'docno': ids[idx], 'text': ' '.join(clean_text[idx])})
#   docs = pd.DataFrame(docs)

#   qrels = []
#   for idx, doc_ids in query_relevant.items():
#     for doc_id in doc_ids:
#       if not pd.isna(doc_id):
#         qrels.append({'qid':idx[-1], 'docno':str(int(doc_id)), 'label':1})
#   qrels = pd.DataFrame(qrels)

#   return docs, queries, qrels

# docs, queries, qrels = prepare_dataset(ids, clean_text, query_list,␣
↪query_relevant)

# # !rm -rf ./pd_index
# # pd_indexer = pt.IterDictIndexer("./pd_index")
# # indexref1 = pd_indexer.index(docs.to_dict(orient="records"))
```

```
[ ]: # !rm -rf ./pretokindex
     # iter_indexer = pt.IterDictIndexer("./pretokindex", threads=1,␣
      ↪pretokenised=True)
     # tok = AutoTokenizer.from_pretrained("bert-base-uncased")
     # token_row_apply = pt.apply.toks(lambda row: Counter(tok.
      ↪tokenize(row['text'])))
     # index_pipe = token_row_apply >> iter_indexer
     # indexref2 = index_pipe.index(docs.to_dict(orient="records"))
```

```
[ ]: # !rm -rf ./colbert_index
     # colbert_indexer = ColBERTIndexer(
     #     checkpoint="http://www.dcs.gla.ac.uk/~craigm/colbert.dnn.zip",
     #     index_root="./colbert_index",
     #     index_name="colbert_index",
     #     chunksize=100,
     #     gpu=True
     # )

     # index = colbert_indexer.index(docs.to_dict(orient="records"))
```

```
[ ]: # import pyterrier_colbert.ranking
     # colbert_factory = pyterrier_colbert.ranking.ColBERTFactory(
     #     "http://www.dcs.gla.ac.uk/~craigm/colbert.dnn.zip", None, None)
     # colbert = colbert_factory.text_scorer(doc_attr='abstract')
```

```
[ ]: # pipeline = pt.terrier.Retriever(index, wmodel='BM25') >> colbert
     # pt.Experiment(
     #     [pipeline],
     #     topics,
     #     qrels,
     #     names=['DPH >> ColBERT'],
     #     eval_metrics=["map", "ndcg", 'ndcg_cut.10', 'P.10', 'mrt']
     # )
```

```
[ ]: # eval_metrics = ['map', 'ndcg_cut_10', 'P_10', 'recall_1000']
     # eval = pt.Experiment([pipeline],
     #                      queries,
     #                      qrels,
     #                      eval_metrics,
     #                      names=['ColBERT'])
```

```
[31]: !pip install nbconvert &> /dev/null
      !apt-get install pandoc &> /dev/null
      !apt-get install texlive-xetex texlive-fonts-recommended texlive-plain-generic␣
       ↪&> /dev/null
```

```
[ ]: !jupyter nbconvert '/content/drive/MyDrive/Colab Notebooks/PartA.ipynb' --to PDF
```