COMP9727 Recommender Systems

Assignment – Content-Based Music Recommendation

Name: Yuhui Wang
zID: z5557746

Part1. Topic Classification

In [7]:
```
import pandas as pd
import nltk
import re
import numpy as np
import matplotlib.pyplot as plt
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
from nltk.stem import PorterStemmer
from sklearn.feature_extraction.text import CountVectorizer, TfidfVectorizer
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, classification_report
from sklearn.naive_bayes import BernoulliNB, MultinomialNB, ComplementNB
from sklearn.model_selection import cross_val_score, cross_validate
from sklearn.metrics.pairwise import cosine_similarity
from sklearn.metrics import precision_score, recall_score, f1_score
```

In [8]:
```
df = pd.read_csv("dataset.tsv", sep='\t')
pd.set_option('display.width', 180)
pd.set_option('display.max_colwidth', 140)
print(df.head(20))
```

```
                             artist_name
track_name  release_date    genre  \
0                                 loving                                        t
he not real lake          2016     rock
1                                incubus
into the summer          2019     rock
2                              reignwolf
hardcore          2016    blues
3                   tedeschi trucks band
anyhow          2016    blues
4   lukas nelson and promise of the real                                        i
f i started over          2017    blues
5                                tia ray
just my luck          2018     jazz
6                             rebelution
trap door          2018   reggae
7                    thank you scientist                             the amateur ars
onist's handbook          2016     jazz
8                             zayde wølf
gladiator          2018     rock
9                         eli young band
never land          2017  country
10                                  klim
ninetofive          2018     jazz
11                          terror squad  rudeboy salute (feat. buju banton, big
pun and fat joe)          2017  hip hop
12                devin townsend project
truth          2016     jazz
13                       vampire weekend
unbearably white          2019     rock
14                                khalid
winter          2017      pop
15                       rend collective                                    countin
g every blessing          2018     rock
16                           post malone                             spoil my night
(feat. swae lee)          2018      pop
17                   albert hammond, jr.                                         ro
cky's late night          2018    blues
18                    the record company
feels so good          2016    blues
19                                  kygo
stole the show          2016      pop


lyrics      topic
0   awake know go see time clear world mirror world mirror magic hour confuse pow
er steal word unheard unheard certain forget bless angry we...       dark
1   shouldn summer pretty build spill ready overflow piss moan ash guess smite le
ave remember call forever shouldn summer summer like coil v...  lifestyle
2   lose deep catch breath think say try break wall mama leave hardcore hardcore
think brother say lock bedroom sister say throw away world ...    sadness
3   run bitter taste take rest feel anchor soul play game rule learn lessons get
choose turn walk away walk away anytime anytime wake feel a...    sadness
4   think think different set apart sober mind sympathetic hearts swear oath swea
r oath know life play play distance great height sure kill ...       dark
5   yeah happen real drink drink turn shots lose count kind luck know friends hou
se sound silly sound stupid sentence break word get fluid l...    emotion
6    long long road occur look shortcut wanna cause scene wanna close okay outspo
ken look trap door sneak grind floor careful wish need answ...       dark
7   quick think good true worst best things forever pessimist drag complicate fee
```

```
l fear tell reason nice slow tell tell reason reason yeah h...       dark
8   start climb face army vipers lions reach cause time tear kingdom liars jail h
eart pessimists nail mouth impressionists spend money thera...       dark
9   word yeah wreck roll lips high good get bottle right right wanna feet cold ha
rd floor kiss steal wanna steal right palm hand fall fall l...    sadness
10  nice place think sky separate nice surprise know life like bring life help ge
ntle kiss good night innocence pray humble amaze beautiful ...   personal
11  grain sand weak blood stain try bottomless hourglass board piece pawn try rea
ch eighth rank sacrifice kings queen field sorrow lose life...       dark
12                                  warn money money money money hallelujah hallelu
jah yeah hello learn live fear peace beauty unfold change home   personal
13  baby pull away unbearably buff mountain sight snow peak unbearably white city
freeze elegant flow wind doorway unbearably cold walk bedr...       dark
14  lose heart nighttime leave cold leave break weary drink lie tell fell morning
get cold life lonely city paso days harder november grow c...   personal
15  blind see colour dead live forever fail redeemer bless measure lose father ch
ange ruin treasure give future bless measure count bless co...   personal
16  come spoil night feelin come play thinkin happen time spoil night spoil night
spoil night night spoil night spoil night necklace natural...  lifestyle
17  darker longer right fear tonight decide everybody feel alright mistake tonigh
t leave companion enjoy polite leave strand confront right ...    sadness
18  wanna right today lookin wanna right today tire bein stun somethin control co
ntrol control taste lips want want want come feel good feel...    emotion
19  darling darling turn light watch watch credit roll cry cry know play house ho
use heroes villains blame wilt roses stage thrill thrill go...    sadness
```

In [9]:

```python
nltk.download('stopwords')
nltk.download('punkt')
nltk.download('punkt_tab')

from sklearn.feature_extraction.text import ENGLISH_STOP_WORDS

ps = PorterStemmer()
stop_words = set(stopwords.words('english'))
sci_stop_words = set(list(ENGLISH_STOP_WORDS)) # scikit-learn stopwords

# Define preprocessing function
def preprocess_text(text):
    text = text.lower()
    text = re.sub(r'[^\w\s]', '', text)     # Keep @, - and '
    tokens = word_tokenize(text)
    tokens = [word for word in tokens if word not in stop_words]
    tokens = [ps.stem(word) for word in tokens]
    return ' '.join(tokens)

def preprocess_text_without_lower(text):
    text = re.sub(r'[^\w\s@\-\']', '', text)     # Keep @, - and '
    tokens = word_tokenize(text)
    tokens = [word for word in tokens if word not in stop_words]
    tokens = [ps.stem(word) for word in tokens]
    return ' '.join(tokens)

def preprocess_text_without_re(text):
    text = text.lower()
    text = re.sub(r'[^\w\s@\-\']', '', text)     # Keep @, - and '
    tokens = word_tokenize(text)
    tokens = [word for word in tokens if word not in stop_words]
    tokens = [ps.stem(word) for word in tokens]
    return ' '.join(tokens)
```

```python
def preprocess_text_with_sci_stopword(text):
    text = text.lower()
    text = re.sub(r'[^\w\s]', '', text)      # Keep @, - and '
    tokens = word_tokenize(text)
    tokens = [word for word in tokens if word not in sci_stop_words]
    tokens = [ps.stem(word) for word in tokens]
    return ' '.join(tokens)

def preprocess_text_without_stopword(text):
    text = text.lower()
    text = re.sub(r'[^\w\s@\-\']', '', text)      # Keep @, - and '
    tokens = word_tokenize(text)
    tokens = [ps.stem(word) for word in tokens]
    return ' '.join(tokens)

def preprocess_text_without_stem(text):
    text = text.lower()
    text = re.sub(r'[^\w\s@\-\']', '', text)      # Keep @, - and '
    tokens = word_tokenize(text)
    tokens = [word for word in tokens if word not in stop_words]
    return ' '.join(tokens)
```

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

Part1-1

(i)the regex might remove too many special characters. I try to keep some special characters like @, -, ' with r'[^\w\s@-']'
(ii) the evaluation is based on only one training test split rather than using cross-validation.I use cross validation and cross_val_score to validate the accuracy later.

In [18]:
```python
df['document'] = df['artist_name'] + ' ' + df['track_name'] + ' ' + df['release_
# Drop duplicates and missing values
df = df.drop_duplicates()
df = df.dropna()
# print(df.info())

# Apply preprocessing to each document
# Test different preprocessing steps
# df['document'] = df['document'].apply(preprocess_text)                    # acc
# df['document'] = df['document'].apply(preprocess_text_without_lower)      # acc
# df['document'] = df['document'].apply(preprocess_text_without_special)    # acc
# df['document'] = df['document'].apply(preprocess_text_without_stopword)   # acc
# df['document'] = df['document'].apply(preprocess_text_with_sci_stopword) # acc
df['document'] = df['document'].apply(preprocess_text_without_stem)      # acc:

# print(df['document'])

vectorizer = CountVectorizer()
X = vectorizer.fit_transform(df['document'])
```

```
# Split the data into training and testing sets
#X_train, X_test, y_train, y_test = train_test_split(X, df['topic'], test_size=0


bnb = BernoulliNB()
# Use cross validation to replace train-test split
cv_scores = cross_val_score(bnb, X, df['topic'], cv=5, scoring='accuracy')
print(f"BNB mean acc: {cv_scores.mean():.3f}" )

mnb = MultinomialNB()
cv_scores = cross_val_score(mnb, X, df['topic'], cv=5, scoring='accuracy')
print(f"MNB mean acc: {cv_scores.mean():.3f}" )
```

```
BNB mean acc: 0.527
MNB mean acc: 0.790
```

Part1-2

According to my experiments above, the best performance (MNB accuracy = 0.79) was achieved **without stemming**. the special characters does not affect the result too much, the result on different stopword lists(NLTK and scikit-learn) are the same, lowcasing has no obvious effect, and not stemming can increase accuracy.

Part1-3

Micro/Macro-Averaging are used for multiple classification. Micro-average is dominated by larger classes and macro-average is dominated by smaller classes which is useful for imbalanced classes. Therefore I use precision_macro and recall_macro here.

The result table shows **MultinomialNB** is better than BernoulliNB.

In [9]:
```
scoring = {
    'accuracy': 'accuracy',
    'precision': 'precision_macro',
    'recall': 'recall_macro'
}
bnb = BernoulliNB()
bnb_scores = cross_validate(bnb, X, df['topic'], cv=5, scoring=scoring)

# print(f"BernoulliNB mean acc: {cv_scores.mean():.3f}" )

mnb = MultinomialNB()
mnb_scores = cross_validate(mnb, X, df['topic'], cv=5, scoring=scoring)
# print(f"MultinomialNB mean acc: {cv_scores.mean():.3f}" )

results = pd.DataFrame({
    'BNB': [bnb_scores['test_accuracy'].mean(),
            bnb_scores['test_precision'].mean(),
            bnb_scores['test_recall'].mean(),],
    'MNB': [mnb_scores['test_accuracy'].mean(),
            mnb_scores['test_precision'].mean(),
            mnb_scores['test_recall'].mean(),]
}, index=['Accuracy', 'Precision', 'Recall'])

print(results.round(4))
```

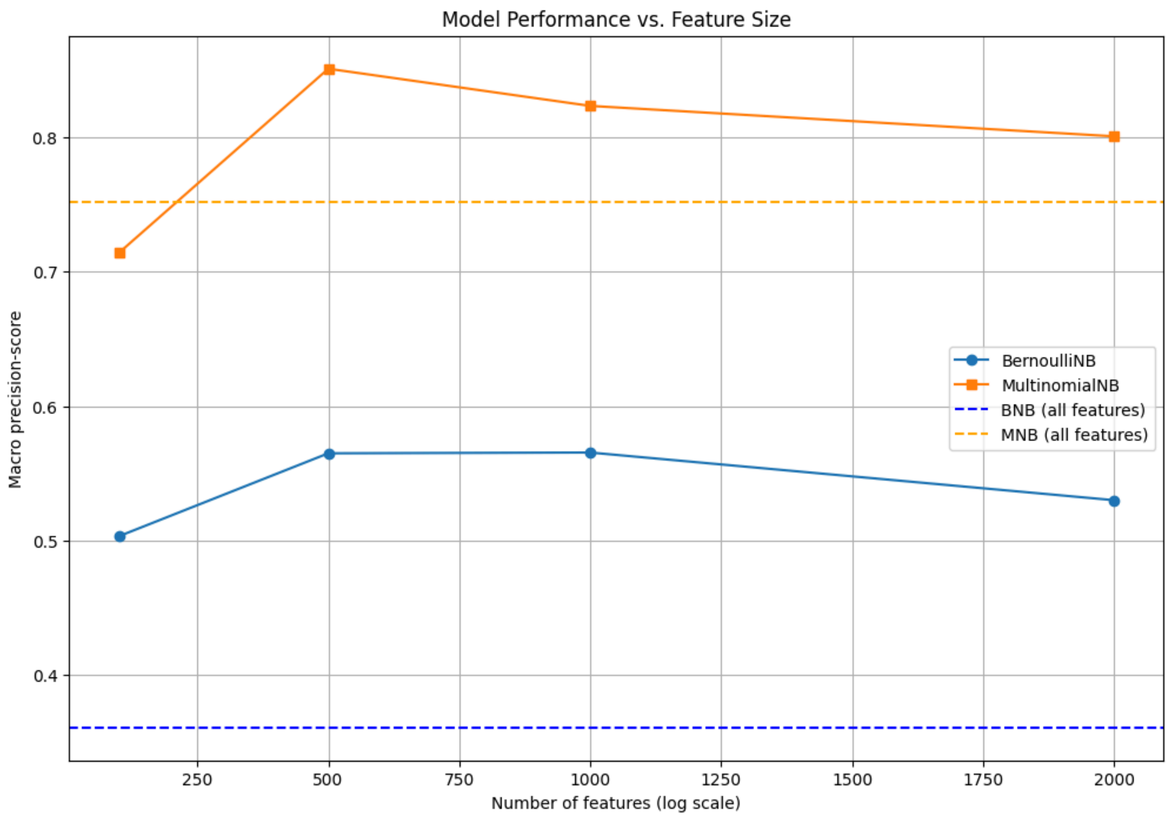
```
             BNB     MNB
Accuracy   0.5270  0.7899
Precision  0.3612  0.7519
Recall     0.3777  0.7015
```

```
/opt/anaconda3/envs/AI/lib/python3.12/site-packages/sklearn/metrics/_classificati
on.py:1706: UndefinedMetricWarning: Precision is ill-defined and being set to 0.0
in labels with no predicted samples. Use `zero_division` parameter to control thi
s behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", result.shape[0])
/opt/anaconda3/envs/AI/lib/python3.12/site-packages/sklearn/metrics/_classificati
on.py:1706: UndefinedMetricWarning: Precision is ill-defined and being set to 0.0
in labels with no predicted samples. Use `zero_division` parameter to control thi
s behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", result.shape[0])
```

In [22]:

```
feature_sizes = [100, 500, 1000, 2000, 'all']
bnb_scores = []
mnb_scores = []

for n_features in feature_sizes:
    vectorizer = CountVectorizer(max_features=None if n_features == 'all' else n
    X = vectorizer.fit_transform(df['document'])
    y = df['topic']
    # Cross-validate (5-fold)
    bnb_score = cross_val_score(bnb, X, y, cv=5, scoring='precision_macro').mean
    mnb_score = cross_val_score(mnb, X, y, cv=5, scoring='precision_macro').mean

    bnb_scores.append(bnb_score)
    mnb_scores.append(mnb_score)

feature_sizes[-1] = len(vectorizer.get_feature_names_out())

plt.figure(figsize=(12, 8))
plt.plot(feature_sizes[:-1], bnb_scores[:-1], 'o-', label='BernoulliNB')
plt.plot(feature_sizes[:-1], mnb_scores[:-1], 's-', label='MultinomialNB')
plt.axhline(y=bnb_scores[-1], color='blue', linestyle='--', label='BNB (all feat
plt.axhline(y=mnb_scores[-1], color='orange', linestyle='--', label='MNB (all fe
# plt.xscale('log')
plt.xlabel('Number of features (log scale)')
plt.ylabel('Macro precision-score')
plt.title('Model Performance vs. Feature Size')
plt.legend()
plt.grid(True)
plt.show()
```

```
/opt/anaconda3/envs/AI/lib/python3.12/site-packages/sklearn/metrics/_classificati
on.py:1706: UndefinedMetricWarning: Precision is ill-defined and being set to 0.0
in labels with no predicted samples. Use `zero_division` parameter to control thi
s behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", result.shape[0])
/opt/anaconda3/envs/AI/lib/python3.12/site-packages/sklearn/metrics/_classificati
on.py:1706: UndefinedMetricWarning: Precision is ill-defined and being set to 0.0
in labels with no predicted samples. Use `zero_division` parameter to control thi
s behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", result.shape[0])
/opt/anaconda3/envs/AI/lib/python3.12/site-packages/sklearn/metrics/_classificati
on.py:1706: UndefinedMetricWarning: Precision is ill-defined and being set to 0.0
in labels with no predicted samples. Use `zero_division` parameter to control thi
s behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", result.shape[0])
```



Part1-4

According to my experiments above, we can see that the best N value is 500 from the graph.

Part1-5

We choose Complement Naive Bayes here. It is said to be an adaption of the standard Multinomial naive Bayes and particularly suited for imbalanced datasets.
We set the max feature to be N=500.
The metrics comparison shows that **MultinomialNB** is better than BernoulliNB and ComplementNB.

In [25]:
```
vectorizer = CountVectorizer(max_features=500)

X = vectorizer.fit_transform(df['document'])
y = df['topic']

bnb = BernoulliNB()
bnb_scores = cross_validate(bnb, X, df['topic'], cv=5, scoring=scoring)

# print(f"BernoulliNB mean acc: {cv_scores.mean():.3f}" )

mnb = MultinomialNB()
mnb_scores = cross_validate(mnb, X, df['topic'], cv=5, scoring=scoring)
# print(f"MultinomialNB mean acc: {cv_scores.mean():.3f}" )

cnb = ComplementNB()
cnb_scores = cross_validate(cnb, X, df['topic'], cv=5, scoring=scoring)
```

```
results = pd.DataFrame({
    'BNB': [bnb_scores['test_accuracy'].mean(),
            bnb_scores['test_precision'].mean(),
            bnb_scores['test_recall'].mean(),],
    'MNB': [mnb_scores['test_accuracy'].mean(),
            mnb_scores['test_precision'].mean(),
            mnb_scores['test_recall'].mean(),],
    'CNB': [cnb_scores['test_accuracy'].mean(),
            cnb_scores['test_precision'].mean(),
            cnb_scores['test_recall'].mean(),],
}, index=['Accuracy', 'Precision', 'Recall'])

print(results.round(4))
```

```
              BNB     MNB     CNB
Accuracy   0.6561  0.8662  0.8608
Precision  0.5649  0.8508  0.8546
Recall     0.5544  0.8451  0.8227
```

In [26]:
```
mnb.fit(X, df['topic'])
```

Out[26]:
```
▾ MultinomialNB
▶ Parameters
```

Part 2. Recommendation Methods

Part2-1

In [34]:
```
df_train = df[:750]
df_test = df[750:1000]
```

In [35]:
```
from collections import defaultdict
topic_to_docs = defaultdict(list)
for idx, row in df_train.iterrows():  # train_data contains Weeks 1-3 (750 songs
    topic_to_docs[row['topic']].append(row['document'])

# topic_vectorizers = {}
topic_matrices = {}
tfidf_vectorizer = TfidfVectorizer(stop_words="english")
X = tfidf_vectorizer.fit_transform(df['document'])

for topic in sorted(topic_to_docs.keys()):
    matrix = tfidf_vectorizer.transform(topic_to_docs[topic])
    topic_matrices[topic] = matrix
    print(f"Topic {topic}: {len(topic_to_docs[topic])} songs, {matrix.shape[1]}
```

```
Topic dark: 246 songs, 9865 features
Topic emotion: 42 songs, 9865 features
Topic lifestyle: 92 songs, 9865 features
Topic personal: 187 songs, 9865 features
Topic sadness: 183 songs, 9865 features
```

In [37]:
```
# Load users interest keywords
def load_user_interests(user_file):
    interests = defaultdict(list)
```

```
    with open(user_file) as f:
        f.readline()
        for line in f:
            topic, keywords = line.strip().split('\t')
            interests[topic] = [kw.lower() for kw in keywords.split()]
    return interests

user1_interests = load_user_interests('user1.tsv')
user2_interests = load_user_interests('user2.tsv')
user3_interests = load_user_interests('user3.tsv')

print(user1_interests)
print(user2_interests)
print(user3_interests)
```

```
defaultdict(<class 'list'>, {'dark': ['fire,', 'enemy,', 'pain,', 'storm,', 'figh
t'], 'sadness': ['cry,', 'alone,', 'heartbroken,', 'tears,', 'regret'], 'persona
l': ['dream,', 'truth,', 'life,', 'growth,', 'identity'], 'lifestyle': ['party,',
'city,', 'night,', 'light,', 'rhythm'], 'emotion': ['love,', 'memory,', 'hug,',
'kiss,', 'feel']})
defaultdict(<class 'list'>, {'sadness': ['lost,', 'sorrow,', 'goodbye,', 'tear
s,', 'silence'], 'emotion': ['romance,', 'touch,', 'feeling,', 'kiss,', 'memor
y']})
defaultdict(<class 'list'>, {'dark': ['"fire,', 'pain,', 'storm,', 'star"'], 'sad
ness': ['"cry,', 'alone,', 'heartbroken,', 'tears,', 'sorrow,', 'goodbye"'], 'per
sonal': ['"dream,', 'truth,', 'life,', 'growth,', 'identity"'], 'lifestyle': ['"p
arty,', 'city,', 'night,', 'light,', 'rhythm"'], 'emotion': ['"love,', 'memory,',
'kiss,', 'hate"']})
```

In [39]:

```
def build_user_profile(data, user_interests, vectorizer, classifier, top_n=10, M
    X = vectorizer.transform(data['document'])
    pred_topics = classifier.predict(X)

    # Filter the songs that users like for each topic
    profile_docs = defaultdict(list)
    for idx, row in data.iterrows():
        if idx == 750:
            break
        pred_topic = pred_topics[idx]

        if pred_topic in user_interests:
            # Calculate the similarity between the song and the user's keywords
            kw_vector = tfidf_vectorizer.transform([' '.join(user_interests.get(
            song_vector = tfidf_vectorizer.transform([row['document']])
            similarity = cosine_similarity(song_vector, kw_vector)[0][0]

            # Record similarity is used for sorting
            profile_docs[pred_topic].append((similarity, row['document']))

    # Select the Top-N songs for each topic and merge the documents
    profile_vectors = {}
    profile_words = {}
    for topic in profile_docs:
        # Sort in descending order of similarity
        sorted_songs = sorted(profile_docs[topic], key=lambda x: x[0], reverse=T
        liked_songs = []
        topic_kws = set(user_interests.get(topic, []))
        # print(f"Topic {topic} kws:", ' '.join(topic_kws))
        for sim, doc in sorted_songs[:top_n]:
            song_words = set(doc.lower().split())
```

```
            if len(topic_kws & song_words) > 0:
                liked_songs.append((sim, doc))

        # Merge the Top-N songs into one document
        combined_doc = ' '.join([doc for (sim, doc) in liked_songs])
        # generate TF-IDF vector
        profile_vectors[topic] = tfidf_vectorizer.transform([combined_doc])

        # print the keywords for each topic
        feature_names = tfidf_vectorizer.get_feature_names_out()
        sorted_idx = np.argsort(profile_vectors[topic].toarray())[0][-M:][::-1]
        top_words = [feature_names[i] for i in sorted_idx]
        profile_words[topic] = top_words
        print(f"【{topic}】profile top 20 words: ")
        print(", ".join(top_words) + "\n")

    return profile_vectors, profile_words

print("User 1 profile:")
user1_profile, user1_words = build_user_profile(df_train, user1_interests, vecto
print("User 2 profile:")
user2_profile, user2_words = build_user_profile(df_train, user2_interests, vecto
print("User 3 profile:")
user3_profile, user3_words = build_user_profile(df_train, user3_interests, vecto
```

```
User 1 profile:
【dark】profile top 20 words:
fight, bleed, gon, na, battle, brother, surround, tell, divide, machine, come, un
ite, white, right, honour, foes, stand, kill, redneck, lead

【lifestyle】profile top 20 words:
sing, song, version, pin, kingdom, tower, rhythm, hour, girl, letter, write, midn
ight, like, kinda, hopeful, favorite, radio, think, loud, bout

【sadness】profile top 20 words:
wish, lay, greater, regret, think, hand, leave, place, beg, wider, cause, blame,
stall, want, tangle, bathroom, head, windows, hold, screen

【emotion】profile top 20 words:
good, feel, lips, doin, kiss, want, coast, right, dirty, know, na, goodbye, hold,
gon, control, south, like, east, baby, uhhuh

【personal】profile top 20 words:
finished, fraction, framework, frame, frailty, frail, fragment, fragile, fracture,
fractal, fossil, foxes, fox, fourth, fountains, fountain, foundation, foul, franc
hiser, frankie

User 2 profile:
【sadness】profile top 20 words:
break, heart, crash, spin, like, cut, dark, apart, save, endless, fall, thunder,
circle, record, deep, hurt, scar, leave, gon, round

【emotion】profile top 20 words:
lips, ease, fade, items, humdrum, onetime, kiss, tingle, autumn, jd, mcpherson, d
eaf, tiny, gleam, charm, fingertips, melodies, like, moonlight, candle

User 3 profile:
【dark】profile top 20 words:
finished, fraction, framework, frame, frailty, frail, fragment, fragile, fracture,
fractal, fossil, foxes, fox, fourth, fountains, fountain, foundation, foul, franc
hiser, frankie

【lifestyle】profile top 20 words:
finished, fraction, framework, frame, frailty, frail, fragment, fragile, fracture,
fractal, fossil, foxes, fox, fourth, fountains, fountain, foundation, foul, franc
hiser, frankie

【sadness】profile top 20 words:
finished, fraction, framework, frame, frailty, frail, fragment, fragile, fracture,
fractal, fossil, foxes, fox, fourth, fountains, fountain, foundation, foul, franc
hiser, frankie

【emotion】profile top 20 words:
finished, fraction, framework, frame, frailty, frail, fragment, fragile, fracture,
fractal, fossil, foxes, fox, fourth, fountains, fountain, foundation, foul, franc
hiser, frankie

【personal】profile top 20 words:
finished, fraction, framework, frame, frailty, frail, fragment, fragile, fracture,
fractal, fossil, foxes, fox, fourth, fountains, fountain, foundation, foul, franc
hiser, frankie
```

Part2-1 Comment

we can see first several keywords in each topic are more reasonable for each topic. This is true for user3, too.

Part2-2

In [41]:
```python
def build_profile_variants(data, interests, vectorizer, classifier, top_n=10, m_
    """ build profiles with different M keywords for each topic"""
    profiles = {}

    # predict topic and filter correct topic songs
    X = vectorizer.transform(data['document'])
    pred_topics = classifier.predict(X)
    # correct_preds = data[pred_topics == data['topic']]

    for m in m_values:
        profile_docs = defaultdict(list)
        for idx, row in data.iterrows():
            if idx == 750:
                break
            pred_topic = pred_topics[idx]

            if pred_topic in interests:
                # Calculate the similarity between the song and the user's keywo
                kw_vector = tfidf_vectorizer.transform([' '.join(interests.get(p
                song_vector = tfidf_vectorizer.transform([row['document']])
                similarity = cosine_similarity(song_vector, kw_vector)[0][0]

                # Record similarity is used for sorting
                profile_docs[pred_topic].append((similarity, row['document']))

        # Select the Top-N songs for each topic and merge the documents
        profile_vectors = {}
        for topic in profile_docs:
            # Sort in descending order of similarity
            sorted_songs = sorted(profile_docs[topic], key=lambda x: x[0], rever
            liked_songs = []
            topic_kws = set(interests.get(topic, []))
            for sim, doc in sorted_songs[:top_n]:
                song_words = set(doc.lower().split())
                if len(topic_kws & song_words) > 0:
                    liked_songs.append((sim, doc))

            # Merge the favorite songs into one document
            combined_doc = ' '.join([doc for (sim, doc) in liked_songs])
            # generate TF-IDF vector
            profile_vectors[topic] = tfidf_vectorizer.transform([combined_doc])

            profiles[f'm={m}'] = profile_vectors
    return profiles

# generate portraits according todifferent M value
profile1_variants = build_profile_variants(df_train, user1_interests, vectorizer
profile2_variants = build_profile_variants(df_train, user2_interests, vectorizer
print(profile1_variants)
```

```
{'m=5': {np.str_('dark'): <Compressed Sparse Row sparse matrix of dtype 'float64'
	with 260 stored elements and shape (1, 9865)>, np.str_('lifestyle'): <Com
pressed Sparse Row sparse matrix of dtype 'float64'
	with 97 stored elements and shape (1, 9865)>, np.str_('sadness'): <Compre
ssed Sparse Row sparse matrix of dtype 'float64'
	with 102 stored elements and shape (1, 9865)>, np.str_('emotion'): <Compr
essed Sparse Row sparse matrix of dtype 'float64'
	with 256 stored elements and shape (1, 9865)>, np.str_('personal'): <Comp
ressed Sparse Row sparse matrix of dtype 'float64'
	with 0 stored elements and shape (1, 9865)>}, 'm=10': {np.str_('dark'): <
Compressed Sparse Row sparse matrix of dtype 'float64'
	with 260 stored elements and shape (1, 9865)>, np.str_('lifestyle'): <Com
pressed Sparse Row sparse matrix of dtype 'float64'
	with 97 stored elements and shape (1, 9865)>, np.str_('sadness'): <Compre
ssed Sparse Row sparse matrix of dtype 'float64'
	with 102 stored elements and shape (1, 9865)>, np.str_('emotion'): <Compr
essed Sparse Row sparse matrix of dtype 'float64'
	with 256 stored elements and shape (1, 9865)>, np.str_('personal'): <Comp
ressed Sparse Row sparse matrix of dtype 'float64'
	with 0 stored elements and shape (1, 9865)>}, 'm=20': {np.str_('dark'): <
Compressed Sparse Row sparse matrix of dtype 'float64'
	with 260 stored elements and shape (1, 9865)>, np.str_('lifestyle'): <Com
pressed Sparse Row sparse matrix of dtype 'float64'
	with 97 stored elements and shape (1, 9865)>, np.str_('sadness'): <Compre
ssed Sparse Row sparse matrix of dtype 'float64'
	with 102 stored elements and shape (1, 9865)>, np.str_('emotion'): <Compr
essed Sparse Row sparse matrix of dtype 'float64'
	with 256 stored elements and shape (1, 9865)>, np.str_('personal'): <Comp
ressed Sparse Row sparse matrix of dtype 'float64'
	with 0 stored elements and shape (1, 9865)>}}
```

Part2-2 Choose N and M value. Use tf-idf vector cos similarity between user profile vectors and song vectors

I choose N=10 for each topic so that there are at most 50 songs for user to choose.

I use precision, recall and f1 as the metrics.

For different M size, we test 5, 10 and 20.

I use tf-idf vector cos similarity between user profile vectors and song vectors for recommending and check the intersection of keywords of each topic with the document words.

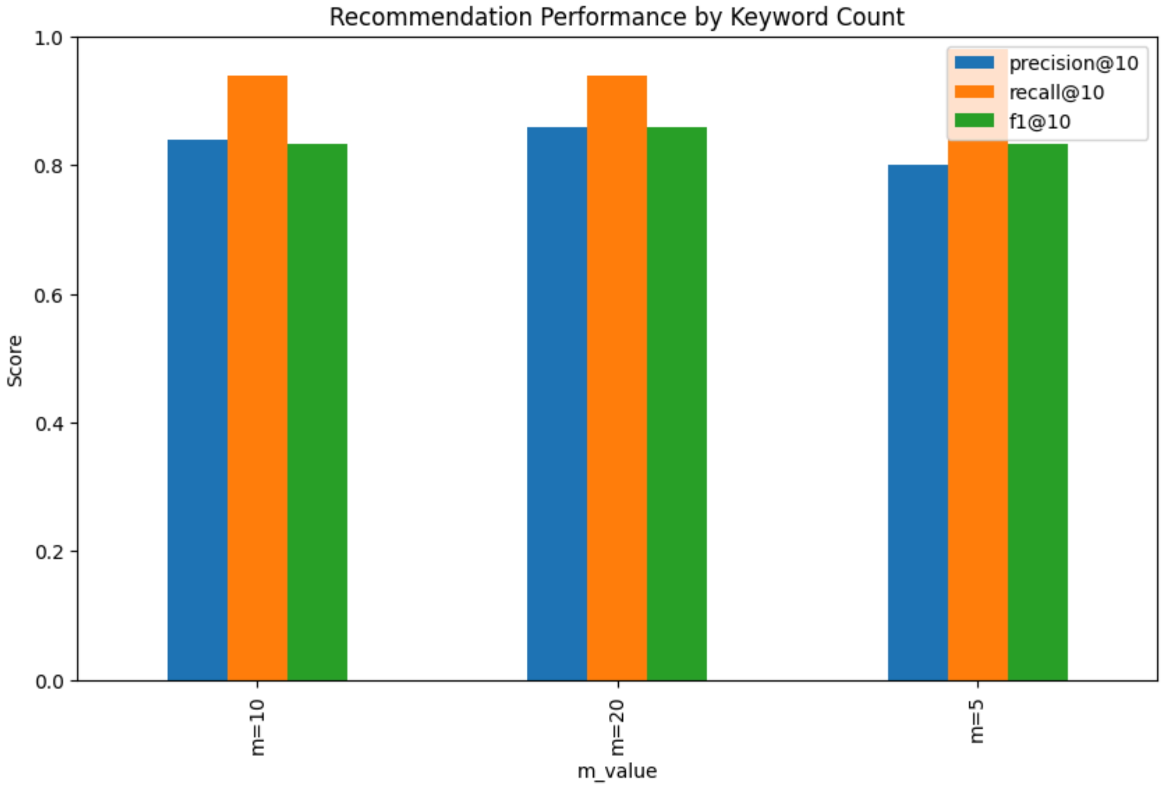
In [43]:

```python
#Simulation Recommendation and Evaluation

def evaluate_recommendations(test_data, profiles, vectorizer, classifier, N=10):
    results = []

    X_test = vectorizer.transform(test_data['document'])
    # print(X_test.shape)
    test_pred_topics = classifier.predict(X_test)
    # print(test_pred_topics.shape)

    for m_name, profile in profiles.items():
        interests = {}
        M = int(m_name[2:])
        print("Test m:", M)
        for topic in profile:
            feature_names = tfidf_vectorizer.get_feature_names_out()
```

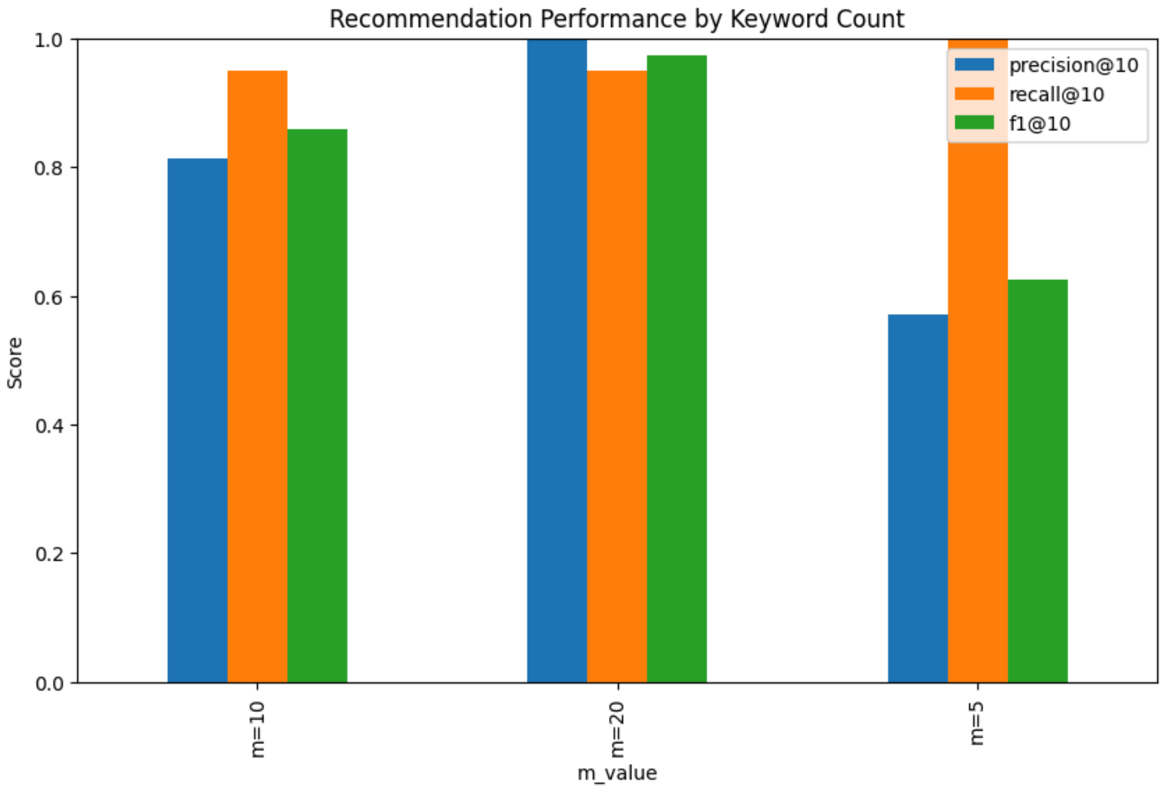
```
            sorted_idx = np.argsort(profile[topic].toarray())[0][-M:][::-1]
            top_words = [feature_names[i] for i in sorted_idx]
            interests[topic] = top_words[:M]

        recommendations = defaultdict(list)
        for i, (_, row) in enumerate(test_data.iterrows()):
            pred_topic = test_pred_topics[i]
            if pred_topic in profile:
                kw_vector = tfidf_vectorizer.transform([' '.join(interests.get(p
                song_vec = tfidf_vectorizer.transform([row['document']])

                similarity = cosine_similarity(song_vec, kw_vector)[0][0]
                recommendations[pred_topic].append((similarity, row))

        # print("M", m_name)
        # Evaluate the recommendations for each topic
        for topic in recommendations:
            # Sort by similarity to take the top 10
            top_songs = sorted(recommendations[topic], key=lambda x: x[0], rever

            # Check whether the user's interest keywords have been hit
            y_true = []
            y_pred = []
            topic_kws = set(interests.get(topic, []))
            # print(f'Topic {topic} kws: {" ".join(topic_kws)}')

            for sim, row in top_songs:
                song_words = set(row['document'].lower().split())
                if len(song_words & topic_kws) > 0:
                    y_true.append(1)
                else:
                    y_true.append(0)

                if topic == row['topic']:
                    y_pred.append(1)
                else:
                    y_pred.append(0)

            if len(y_true) > 0:
                precision = precision_score(y_true, y_pred, zero_division=0)
                recall = recall_score(y_true, y_pred, zero_division=0)
                f1 = f1_score(y_true, y_pred, zero_division=0)

                results.append({
                    'm_value': m_name,
                    'topic': topic,
                    'precision@10': precision,
                    'recall@10': recall,
                    'f1@10': f1,
                    # 'recommendations': [s[1]['document'] for s in top_songs]
                })

    return pd.DataFrame(results)

eval1_results = evaluate_recommendations(df_test, profile1_variants, vectorizer,
eval2_results = evaluate_recommendations(df_test, profile2_variants, vectorizer,

# 7. analyze the result and print metrics
def analyze_results(results_df):
    summary = results_df.groupby('m_value')[['precision@10','recall@10','f1@10']
```

```
    print("=== metrics ===")
    print(summary.sort_values('precision@10', ascending=False))

    summary.plot(kind='bar', figsize=(10,6))
    plt.title('Recommendation Performance by Keyword Count')
    plt.ylabel('Score')
    plt.ylim(0, 1)
    plt.show()

    return summary

performance_summary = analyze_results(eval1_results)
performance_summary = analyze_results(eval2_results)
```

```
Test m: 5
Test m: 10
Test m: 20
Test m: 5
Test m: 10
Test m: 20
=== metrics ===
         precision@10  recall@10     f1@10
m_value
m=20             0.86       0.94  0.859559
m=10             0.84       0.94  0.833918
m=5              0.80       0.98  0.833918
```




```
=== metrics ===
         precision@10  recall@10     f1@10
m_value
m=20         1.000000       0.95  0.973684
m=10         0.812500       0.95  0.858300
m=5          0.571429       1.00  0.625000
```



Part 3. User Evaluation

In [44]:
```
sample1 = df_train[0:250].sample(n=10, random_state=42)
sample2 = df_train[250:500].sample(n=10, random_state=42)
sample3 = df_train[500:750].sample(n=10, random_state=42)
print(sample1['document'])
```

```
142    skool 77 vivo hip hop live 2017 hip hop head camden sure raise toast patro
n saint waifs stray bingham ghost freshfaced lass kentish come...
6      rebelution trap door 2018 reggae long long road occur look shortcut wan na
cause scene wan na close okay outspoken look trap door sneak ...
97     alec benjamin outrunning karma 2018 pop outrun karma charmer bug larva fol
low colorado dozen hearts body lie drag colorado modern desper...
60     phish come outlive brains 2018 blues overlap future pass pass vapor light
liquid blue look exactly suppose look shape hang look shape ha...
112    madeleine peyroux shout sister shout 2016 jazz shout sister shout halleluj
ah shout sister shout shout sister shout tell world reason liv...
181    janiva magness could 2018 blues pay dues truth leave leave leave forget si
lence rule blind fool learn learn learn break away rise days f...
197    eric ethridge met first 2018 country strangest creature fail prey like vul
ture hand follow follow follow life steal light sweet taste ta...
184    imagine dragons natural 2018 rock hold line give give tell house come cons
equence cost tell star align heaven step save cause house stan...
9      eli young band never land 2017 country word yeah wreck roll lips high good
get bottle right right wan na feet cold hard floor kiss steal...
104    larkin poe john revelator 2017 blues tell write revelator tell write revel
ator tell write revelator write book seven seal tell write rev...
Name: document, dtype: object
```

In [45]:
```
likes1 = [
    0,
    0,
    0,
    1,
```

```
    0,
    1,
    0,
    1,
    0,
    0
]
```

In [46]:
```
print(sample2['document'])
```

```
392    jon bellion stupid deep 2018 rock hop fight feel free things yeah attempt
earn yeah cause hole inside heart stupid deep stupid deep try ...
256    kills black tar 2016 blues invention handsome fairytales fair game world l
ook sharpen blade london blood thirsty paris vein open vein pu...
347    haken good doctor 2018 jazz call cell block nurse inmates scream bed silen
t unusual delude psychotic catatonic good doctor look smile ti...
310    allen toussaint american tune 2016 blues time mistake time confuse fall fo
rsake certainly misuse alright alright weary bone expect brigh...
362    tenth avenue north hope 2016 rock walk great unknown question come questio
n purpose pain tear vain want live fear want trust near trust ...
431    surfaces heaven falls fall 2018 pop wake early mornin feel light open wind
ow shadow banana pancakes problems jam johnson swear hear call...
447    blues saraceno devils got beat 2019 blues evil mind trouble feet live devi
l beat evil mind trouble feet live devil devil beat walk cours...
434    wood brothers 2018 blues things think miss miss miss dream free miss chang
e world change world need things think know things know know c...
259    mild high club tesselation 2016 rock daddy know hand pool head fool push w
eight hollywood gestalt hang edge fault learn trick uncle educ...
354    parov stelar snake charmer 2019 jazz hear music away party night walk city
streets bump blow whistle everyday hear time change look eye ...
Name: document, dtype: object
```

In [47]:
```
likes2 = [
    0,
    0,
    0,
    0,
    1,
    1,
    0,
    1,
    0,
    0
]
```

In [48]:
```
print(sample3['document'])
```

```
643    score fear 2019 rock whoaohoh whoaohoh whoaohoh whoaohoh whoaohoh whoaohoh
whoaohoh whoaohoh knock demons creep round change shots runni...
506    runaway june buy drinks 2019 pop yeah try unfall apart think neon light re
al good start call couple friends stay guess go heart break me...
597    goodbye june get happy 2018 blues begin yellow skin black night drink till
light cocaine closest distance line mind try erase past ways ...
560    avril lavigne head water 2019 pop got ta calm want want windows doors safe
warm yeah life fight reach shore voice drive force pull overb...
613    shenseea tie 2018 reggae shenseea oouuuu oouuuu oouuu oouuu strangle viagr
a chiney brush tight underneath clean tidy cover mouth whine h...
683    circa waves times wont change 2019 rock hear come thunder wonder fall morn
ing time change time change change corner label neck come save...
699    iya terra livication 2017 reggae livication good right say mountains organ
isms humanity learn free feel soul learn almighty hold know ra...
686    fall boy church 2018 rock church knees confess know sanctuary holy church
knees knees knees knees pain billboard swallow time capsule fu...
509    shane shane youre worthy 2017 rock sing song lift voice dark sing sing lif
t eye stand sing hop fear want years life resolve sing song li...
605    blues saraceno dark horse always wins 2018 blues deliver evil deliver deli
ver deliver evil yeah deliver oooh oooh dark horse win oooh oo...
Name: document, dtype: object
```

In [49]:

```
likes3 = [
    0,
    1,
    1,
    0,
    0,
    0,
    0,
    0,
    0,
    0
]
```

In [50]:

```
liked_songs = []
for i, (_, row) in enumerate(sample1.iterrows()):
    if likes1[i] == 1:
        liked_songs.append(row['document'])

for i, (_, row) in enumerate(sample2.iterrows()):
    if likes2[i] == 1:
        liked_songs.append(row['document'])

for i, (_, row) in enumerate(sample3.iterrows()):
    if likes3[i] == 1:
        liked_songs.append(row['document'])

combined_document = ' '.join(liked_songs)
print(combined_document)
```

```
phish come outlive brains 2018 blues overlap future pass pass vapor light liquid
blue look exactly suppose look shape hang look shape hang look stop try rush natu
re slow come outlive brain distance frown come outlive brain distance frown come
outlive brain distance frown come outlive brain distance frown cub cub cub cub cu
b cub cub cub cub cub cub cub glue magnet glue magnet glue magnet glue magnet glu
e magnet glue magnet glue magnet glue magnet glue magnet glue magnet glue magnet
glue glue magnet glue magnet glue magnet glue magnet glue magnet come outlive bra
in come outlive brain come outlive brain come outlive brain come outlive brain ja
niva magness could 2018 blues pay dues truth leave leave leave forget silence rul
e blind fool learn learn learn break away rise days firelight fade night estrange
draw flame like rain strange go away go away pay dues truce leave leave leave for
get rule truth learn learn learn break away rise days firelight fade night estran
ge draw flame like rain strange strange days firelight fade night estrange draw f
lame like rain strange go away away imagine dragons natural 2018 rock hold line g
ive give tell house come consequence cost tell star align heaven step save cause
house stand strong leave heart cast away product today prey stand edge face cause
natural beat heart stone got ta cold world yeah natural live life cutthroat got t
a cold yeah natural lyric commercial tenth avenue north hope 2016 rock walk great
unknown question come question purpose pain tear vain want live fear want trust n
ear trust see triumph tragedy depth soul flood feel like night catch tear leave d
epth soul flood depth soul flood happen afraid cause closer breath calm hear beli
eve face depth soul flood depth soul flood flood flood surfaces heaven falls fall
2018 pop wake early mornin feel light open window shadow banana pancakes problems
jam johnson swear hear call outside heaven fall heaven fall heaven faaaalls heave
n fall heaven fall heaven faaall nothin like color paint better good time horizon
couple bird come fee swear hear call outside heaven fall heaven fall heaven faaaa
lls heaven fall heaven fall heaven faaall heaven fall heaven fall heaven faaaalls
heaven fall heaven fall heaven faaall fall fall fall fall fall fall fall fall woo
d brothers 2018 blues things think miss miss miss dream free miss change world ch
ange world need things think know things know know change world change world need
wan na know eitheror yeah things think miss things think miss change world change
world change world change world need need need runaway june buy drinks 2019 pop y
eah try unfall apart think neon light real good start call couple friends stay gu
ess go heart break mean got ta stay home yeah drink night light drop change jukeb
ox dance stop thinkin bout drinkin bout need yeah drink tonight tonight tonight y
eah drink tonight tonight tonight dive type walk see drink hand yeah say sweet gl
ass gon na pass time yeah fine drink night light drop change jukebox dance stop t
hinkin bout drinkin bout need yeah drink tonight tonight tonight yeah drink tonig
ht tonight tonight walk self door self medicate headache yeah boyfriend drink nig
ht light drop change jukebox dance time stop stop thinkin bout drinkin bout need
yeah drink tonight tonight tonight yeah drink tonight tonight tonight drop change
jukebox tonight tonight tonight thinkin bout drinkin bout dance tonight tonight t
onight goodbye june get happy 2018 blues begin yellow skin black night drink till
light cocaine closest distance line mind try erase past ways kill pain maybe pill
s alcohol weed shake memories try therapy gon na bring check doubt maybe pills al
cohol weed shake memories think need sleep live dream high like anxiety change ba
by believe shake shake shake memories try forgive lose pain okay try forgive try
bottle shelf know help till
```

In [51]:
```python
sample_df = pd.concat([sample1, sample2,sample3], ignore_index=True)

X_test = vectorizer.transform(sample_df['document'])
pred_topics = mnb.predict(X_test)

profile_docs = defaultdict(list)
for idx, row in sample_df.iterrows():
    pred_topic = pred_topics[idx]
    profile_docs[pred_topic].append(row['document'])

profile_vectors = {}
```

```python
for topic in profile_docs:
    liked_songs = []
    for doc in profile_docs[topic]:
        liked_songs.append(doc)

    # 合并喜欢的歌曲为一个文档
    combined_doc = ' '.join([doc for doc in liked_songs])
    # 生成TF-IDF向量
    profile_vectors[topic] = tfidf_vectorizer.transform([combined_doc])

results = []
interests = {}
M=10
N=10
for topic in profile_vectors:
    feature_names = tfidf_vectorizer.get_feature_names_out()
    sorted_idx = np.argsort(profile_vectors[topic].toarray())[0][-M:][::-1]
    top_words = [feature_names[i] for i in sorted_idx]
    interests[topic] = top_words[:M]

recommendations = []
for i, (_, row) in enumerate(sample_df.iterrows()):
    pred_topic = pred_topics[i]
    if pred_topic in profile_vectors:
        kw_vector = tfidf_vectorizer.transform([' '.join(interests.get(pred_topi
        song_vec = tfidf_vectorizer.transform([row['document']])
        similarity = cosine_similarity(song_vec, kw_vector)[0][0]
        recommendations.append((similarity, row))

# show N songs
top_songs = sorted(recommendations, key=lambda x: x[0], reverse=True)[:N]

for i, (sim, row) in enumerate(top_songs):
    print(f"{i}# {row['artist_name']}: [{row['track_name']}], {row['genre']}: {r
```

0# larkin poe: [john the revelator], blues: tell write revelator tell write revelator tell write revelator write book seven seal tell write revelator tell write revelator tell write revelator write book seven seal walk cool call refuse answer cause naked ashamed tell write revelator tell write revelator tell write revelator write book seven seal apostles away say watch hour till yonder pray tell write revelator tell write revelator tell write revelator write book seven seal tell write revelator tell write revelator tell write revelator write book seven seal yeah tell write revelator tell write revelator tell write revelator write book seven seal write book seven seal write book seven seal
1# the wood brothers: [this is it], blues: things think miss miss shouldn miss dream free miss change world change world need things think know things know know change world change world need wanna know eitheror yeah things think miss things think miss change world change world change world change world need need need
2# parov stelar: [snake charmer], jazz: hear music away party night walk city streets bump blow whistle everyday hear time change look eye think hypnotize feel dance time play put trance black mamba lady slowly come hide basket music stop mister snake charmer play flute crawl play tune snake charmer play flute crawl play tune snake charmer play flute crawl play tune snake charmer snake tattoo tell want want think hypnotize feel dance time play put trance black mamba lady slowly come hide basket music stop dance kill venom think escape tongue swallow piece completely numb look charmer play song mister snake charmer play flute crawl play tune snake charmer play flute crawl play tune snake charmer play flute crawl play tune snake charmer snake tattoo tell want want want mister snake charmer play flute crawl play tune snake charmer snake tattoo tell want want
3# haken: [the good doctor], jazz: call cell block nurse inmates scream bed silent unusual delude psychotic catatonic good doctor look smile time game electricity prescription need bring society electricity cure need bring empire knees inside mind spark vague memories cave break life inside mind spark vague memories cave break life electricity prescription need bring society electricity cure need bring empire knees sure arm bind pills secrets drown render mind unsound inside mind spark vague memories cave break life inside mind inside mind spark spark vague memories cave break life electricity prescription need bring society electricity cure need bring empire knees
4# phish: [we are come to outlive our brains], blues: overlap future pass pass vapor light liquid blue look exactly suppose look shape hang look shape hang look stop try rush nature slow come outlive brain distance frown come outlive brain distance frown come outlive brain distance frown come outlive brain distance frown cub cub cub cub cub cub cub cub cub cub cub cub glue magnet glue magnet glue magnet glue magnet glue magnet glue magnet glue magnet glue magnet glue magnet glue magnet glue magnet glue glue magnet glue magnet glue magnet glue magnet glue magnet come outlive brain come outlive brain come outlive brain come outlive brain come outlive brain
5# allen toussaint: [american tune], blues: time mistake time confuse fall forsake certainly misuse alright alright weary bone expect bright bear divine away home away home know soul better friend feel ease know dream shatter drive knees alright alright live long think road travel wonder go wrong help wonder go wrong dream die dream soul unexpectedly look smile assuredly dream fly high eye clearly statue liberty sail away dream fly come ship mayflower come ship sail moon come hat uncertain hours sing american tune alright alright alright forever bless tomorrow gonna work try rest try rest alright alright alright forever bless tomorrows gonna work try rest try rest
6# eli young band: [never land], country: word yeah wreck roll lips high good get bottle right right wanna feet cold hard floor kiss steal wanna steal right palm hand fall fall like land head heartbeat hit feel like gravity exist wanna stop know fall fall like land hang like dance stretch song long float like make memories night night wanna feet cold hard floor kiss steal wanna steal right palm hand fall fall like land head heartbeat hit feel like gravity exist wanna stop know fall fall like land fall like land stay get fly wanna feet cold hard floor kiss steal wanna steal right palm hand fall fall like land head heartbeat hit feel like gravity exist wanna stop know fall fall like land fall like land fall like land fall

```
fall fall like land fall fall fall like land fall fall fall like land
7# madeleine peyroux: [shout sister shout], jazz: shout sister shout hallelujah s
hout sister shout shout sister shout tell world reason live reason die darn good
reason woman start cry reason mole reason dimple reason simple shout sister shout
hallelujah shout sister shout hallelujah shout sister shout mmhmm yeah shout sist
er shout tell world brilliant fool heaven observe golden rule sweetheart wife qui
t baby lose life shout sister shout hallelujah shout sister shout hallelujah shou
t sister shout yeah tell world reason mountain reason reason doctor give patient
pill reason dance reason sing reason band swing understand bird understand be und
erstand eat cheese understand understand cat rhythm understand shout sister shout
hallelujah shout sister shout hallelujah shout sister shout mmhmm hallelujah tell
world know gonna shout shout sister go shout shout sister go shout tell world
8# goodbye june: [get happy], blues: begin yellow skin black night drink till lig
ht cocaine closest distance line mind try erase past ways kill pain maybe pills a
lcohol weed shake memories try therapy gonna bring check doubt maybe pills alcoho
l weed shake memories think need sleep live dream high like anxiety change baby b
elieve shake shake shake memories try forgive lose pain okay try forgive try bott
le shelf know help till
9# shenseea: [tie me up], reggae: shenseea oouuuu oouuuu oouuu oouuu strangle via
gra chiney brush tight underneath clean tidy cover mouth whine hand throat tight
pumpum clean tidy yeah ready girl baby beat like hate bawl lord come save like sl
avery turn steer leave right shift stick gear grind hand breast like cement pave
hotter hell like sinner hold squeeze like trigger lack cause blurry vision come g
low shimmer like bimma oouuu oouuu oouuu oouuu oouuu oouuu uptie oouuu ttie stran
gle viagra chiney brush tight underneath clean tidy cover mouth whine hand throat
tight pumpum clean tidy  lock like charge felony ball swing like  legacy speciall
y long like line embassy spank cause like stop touch right spot submissive babes
fight pass cloud like pilot oouuu oouuu oouuu oouuu oouuu oouuu uptie oouuu ttie
strangle viagra chiney brush tight underneath clean tidy cover mouth whine hand t
hroat tight pumpum clean tidy
```

In [52]:

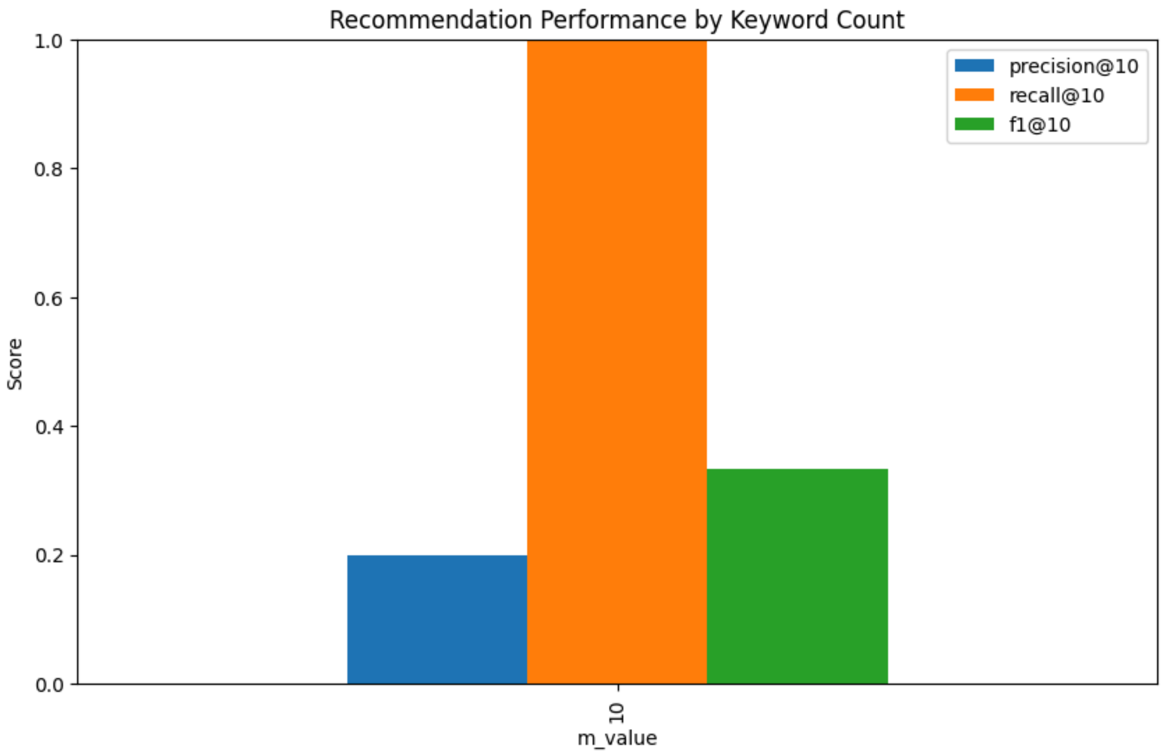
```python
# user feedback
likes = [
    0,
    1,
    0,
    0,
    0,
    0,
    1,
    0,
    0,
    0
]
```

In [54]:

```python
# Check if the user's interest keywords have been hit

y_true = likes
y_pred = [1 for i in range(N)]

if len(y_true) > 0:
    precision = precision_score(y_true, y_pred, zero_division=0)
    recall = recall_score(y_true, y_pred, zero_division=0)
    f1 = f1_score(y_true, y_pred, zero_division=0)

    results.append({
        'm_value': M,
        'topic': topic,
        'precision@10': precision,
        'recall@10': recall,
```

```
        'f1@10': f1,
    })

eval_results = pd.DataFrame(results)
performance_summary = analyze_results(eval_results)
```

```
=== metrics ===
         precision@10  recall@10     f1@10
m_value
10                0.2        1.0  0.333333
```



Part3-Conclusion and real user's feedback

For real user, the precision@10 and f1@10 is much lower than imagined user, because the taste of real user is not very stable when he just look at the songs across songs of different topics.
My user's feedback: the recommendation is a little hard to use, but I can see it needs a lot of hard work. I tried to choose some songs I may like, but mostly I'm not very interested.