ZID: Z5532140

Full Name: Haocheng Yang

Part 1

Part1 Q1

Part 1 - Question 1: Cleaning and Evaluation Fixes

1. **Over-aggressive Regex Cleaning**

It will remove all non-letter characters, including useful punctuation, numbers, emojis, musical genres such as "R&B", hyphens of "hip-hop", etc. **Fix**: I use a single revised regular expression `[^a-zA-Z0-9\s]` to retain letters, numbers, hyphens, and apostrophes. It keeps meaningful phrases in lyrics and metadata.

2. **Single Train-Test Split Instead of Cross-Validation**

The tutorial uses a single random split, and this is not a good one for performance estimation.

Solution: I use **5-fold cross-validation** via `cross_val_score` or `cross_validate`. This provides a more consistent and representative evaluation.

Part1 Q2

Data loading

In [1]:
```
import pandas as pd

df = pd.read_csv("dataset.tsv", sep="\t")

df["document"] = df["artist_name"].fillna("") + " " + \
                 df["track_name"].fillna("") + " " + \
                 df["genre"].fillna("") + " " + \
                 df["lyrics"].fillna("")

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

Out[1]:
```
topic
dark         490
sadness      376
personal     347
lifestyle    205
emotion       82
Name: count, dtype: int64
```

Text cleaning and preprocessing functions

In [2]:
```
import re
import nltk
from nltk.corpus import stopwords
from nltk.stem import PorterStemmer
from nltk.tokenize import word_tokenize
from sklearn.naive_bayes import MultinomialNB
from nltk.stem import WordNetLemmatizer
from sklearn.feature_extraction.text import ENGLISH_STOP_WORDS

nltk.download('punkt')
nltk.download('stopwords')
nltk.download('wordnet')

stemmer = PorterStemmer()
lemmatizer = WordNetLemmatizer()
nltk_stop_words = set(stopwords.words('english'))
sklearn_stopwords = ENGLISH_STOP_WORDS

def preprocess(text, use_nltk=False, use_scikit_learn=False, use_stem=False, use
    if pd.isna(text):
            return ''

    text = text.lower()

    text = re.sub(r'\s+', ' ', text)

    text = re.sub(r'[^a-zA-Z0-9\s]', '', text)

    text = re.sub(r'\s+', ' ', text)

    text = text.strip()

    tokens = word_tokenize(text)

    if use_nltk:
        tokens = [t for t in tokens if t not in nltk_stop_words]
    if use_scikit_learn:
        tokens = [t for t in tokens if t not in sklearn_stopwords]

    if use_stem:
        tokens = [stemmer.stem(t) for t in tokens]
    if use_lemma:
        tokens = [lemmatizer.lemmatize(t) for t in tokens]

    return " ".join(tokens)
```

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

Define the preprocessing combination

In [3]:
```
preprocessing_configs = {
    'base (lower + nltk_stopwords)': lambda x: preprocess(x, True, False, False,
    'base (lower + sklearn_stopwords)': lambda x: preprocess(x, False, True, Fal

    'stem + nltk_stopwords': lambda x: preprocess(x, True, False, True, False),
    'stem + sklearn_stopwords': lambda x: preprocess(x, False, True, True, False

    'lemma + nltk_stopwords': lambda x: preprocess(x, True, False, False, True),
    'lemma + sklearn_stopwords': lambda x: preprocess(x, False, True, False, Tru

    'stem + lemma + nltk_stopwords': lambda x: preprocess(x, True, False, True,
    'stem + lemma + sklearn_stopwords': lambda x: preprocess(x, False, True, Tru
}
```

Evaluate the preprocessing combination

In [4]:
```
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.model_selection import cross_val_score

# MultinomialNB + CountVectorizer + 5-fold
def evaluate_pipeline(text_series, labels, preproc_func):
    cleaned = text_series.apply(preproc_func)
    X = CountVectorizer().fit_transform(cleaned)
    y = labels
    model = MultinomialNB()
    scores = cross_val_score(model, X, y, cv=5, scoring='accuracy')
    return scores.mean()

results = []
for name, func in preprocessing_configs.items():
    print(f"Evaluating: {name}")
    acc = evaluate_pipeline(df["document"], df["topic"], func)
    results.append((name, acc))

df_results = pd.DataFrame(results, columns=["Preprocessing Strategy", "Average A
df_results = df_results.sort_values(by="Average Accuracy", ascending=False)
print(df_results.to_string(index=False))
```

```
Evaluating: base (lower + nltk_stopwords)
Evaluating: base (lower + sklearn_stopwords)
Evaluating: stem + nltk_stopwords
Evaluating: stem + sklearn_stopwords
Evaluating: lemma + nltk_stopwords
Evaluating: lemma + sklearn_stopwords
Evaluating: stem + lemma + nltk_stopwords
Evaluating: stem + lemma + sklearn_stopwords
          Preprocessing Strategy  Average Accuracy
   stem + lemma + nltk_stopwords          0.795333
            stem + nltk_stopwords          0.794667
           lemma + nltk_stopwords          0.791333
   base (lower + nltk_stopwords)          0.789333
base (lower + sklearn_stopwords)          0.786667
        stem + sklearn_stopwords          0.784000
stem + lemma + sklearn_stopwords          0.784000
       lemma + sklearn_stopwords          0.783333
```

Different Naive Bayes preprocessing methods were compared using five cross-validations. 'stem + lemma + nltk_stopwords' was best with being 79.53%.

For the current task, lemmatization can retain more semantic consistency and improve the matching quality

Therefore, I choose **'stem + lemma + nltk_stopwords'** as the best preprocessing scheme. This is the best compromise between performance and simplicity.

Part 1 Q3

Import and Preparation

In [5]:
```
from sklearn.model_selection import cross_validate
from sklearn.naive_bayes import BernoulliNB, MultinomialNB
import numpy as np
import pandas as pd

df["clean_text"] = df["document"].apply(lambda x: preprocess(x, True, False, Tru
vectorizer = CountVectorizer()
X = vectorizer.fit_transform(df["clean_text"])
y = df["topic"]
```

Cross-validation + multi-metric evaluation

In [6]:
```
from sklearn.metrics import precision_score, recall_score, f1_score, make_scorer

scoring = {
    'accuracy': 'accuracy',
    'precision_macro': make_scorer(precision_score, average='macro', zero_divisi
    'recall_macro': make_scorer(recall_score, average='macro', zero_division=0),
    'f1_macro': make_scorer(f1_score, average='macro', zero_division=0)
}

bnb = BernoulliNB()
mnb = MultinomialNB()
```

```
bnb_scores = cross_validate(bnb, X, y, cv=5, scoring=scoring)
mnb_scores = cross_validate(mnb, X, y, cv=5, scoring=scoring)
```

Result summary and presentation

In [7]:
```
results_df = pd.DataFrame({
    "Metric": ['Accuracy', 'Macro Precision', 'Macro Recall', 'Macro F1'],
    "BernoulliNB": [
        np.mean(bnb_scores['test_accuracy']),
        np.mean(bnb_scores['test_precision_macro']),
        np.mean(bnb_scores['test_recall_macro']),
        np.mean(bnb_scores['test_f1_macro'])
    ],
    "MultinomialNB": [
        np.mean(mnb_scores['test_accuracy']),
        np.mean(mnb_scores['test_precision_macro']),
        np.mean(mnb_scores['test_recall_macro']),
        np.mean(mnb_scores['test_f1_macro'])
    ]
})

print("=== BNB vs MNB Classification Metrics ===")
print(results_df)
```

```
=== BNB vs MNB Classification Metrics ===
            Metric  BernoulliNB  MultinomialNB
0         Accuracy     0.524000       0.795333
1  Macro Precision     0.384789       0.768120
2     Macro Recall     0.382248       0.713958
3         Macro F1     0.342412       0.729234
```

The basis for comparison and selection of model evaluation metrics

To compare BNB and MNB's performance in the task of "topic classification", I selected the following four indicators of evaluation:

Accuracy : It determines the proportion of correct predictions by the overall model. Assuming that category distribution is roughly balanced, it is a good evaluation metric.

Macro Precision : Calculate the precision of each category separately and then take the average value. This metric gives each category equal weight, which can prevent the model from being too biased toward large categories.

Macro Recall : Calculate the recall rate for each category separately and then take the average. This metric helps in calculating whether the model is able to cover samples of all categories or not.

Macro F1 score : Harmonic mean of precision and recall, it is used to measure the overall robustness of the model, especially suitable for multi-category tasks.

Get by checking the number of different types of samples by `df["topic"].value_counts()` :

dark: 490

sadness: 376

personal: 347

lifestyle: 205

emotion: 82

Among them, the emotion category is significantly fewer. Therefore, if the accuracy rate is used alone, the model would be willing to favor the more frequently predicted category and neglect the minority ones. In short, although the accuracy rate can be used as a general benchmark, I mainly rely on the `Macro-F1 score` to judge whether the final model is better or worse than others.

It can be seen from the BNB vs MNB Classification Metrics that in all of the metrics used for evaluation, MultinomialNB performs better than BernoulliNB.

Part 1 Q4

Define the variable vectorizer and the evaluation function

In [8]:
```
from sklearn.metrics import confusion_matrix, classification_report
def evaluate_with_metrics(text_series, labels, max_features, isMnb):
    vectorizer = CountVectorizer(max_features=max_features)
    X = vectorizer.fit_transform(text_series)

    if isMnb:
        model = MultinomialNB()
    else:
        model = BernoulliNB()

    scoring = {
        'accuracy': 'accuracy',
        'precision_macro': make_scorer(precision_score, average='macro', zero_di
        'recall_macro': make_scorer(recall_score, average='macro', zero_division
        'f1_macro': make_scorer(f1_score, average='macro', zero_division=0)
    }

    scores = cross_validate(model, X, labels, cv=5, scoring=scoring)

    model.fit(X, labels)
    y_pred = model.predict(X)
    print("Confusion Matrix:\n", confusion_matrix(labels, y_pred, labels=model.c
    print("Classification Report:\n", classification_report(labels, y_pred))

    return {
        'accuracy': scores['test_accuracy'].mean(),
        'precision': scores['test_precision_macro'].mean(),
        'recall': scores['test_recall_macro'].mean(),
        'f1': scores['test_f1_macro'].mean()
    }
```

Experiment with multiple values of N and collect the results

In [9]:
```
N_values = [200, 300, 400, 500, 1000, 2000, 3000, 5000, 7000, 10000, 15000, None
results_Mnb = []
results_Bnb = []

for n in N_values:
    print(f"Evaluating MNB max_features = {n}")
    acc_Mnb = evaluate_with_metrics(df["clean_text"], df["topic"], max_features=
    label_Mnb = "No Limit" if n is None else n
    results_Mnb.append((label_Mnb, acc_Mnb))


    print(f"Evaluating BNB max_features = {n}")
    acc_Bnb = evaluate_with_metrics(df["clean_text"], df["topic"], max_features=
    label_Bnb = "No Limit" if n is None else n
    results_Bnb.append((label_Bnb, acc_Bnb))

results_df_Mnb = pd.DataFrame(results_Mnb, columns=["Top-N Features", "Accuracy"
results_df_Bnb = pd.DataFrame(results_Bnb, columns=["Top-N Features", "Accuracy"
```

```
Evaluating MNB max_features = 200
Confusion Matrix:
 [[417   8   9  25  31]
 [  4  67   3   1   7]
 [ 14   2 184   3   2]
 [ 23   3   4 310   7]
 [ 15   1   5  11 344]]
Classification Report:
               precision    recall  f1-score   support

        dark       0.88      0.85      0.87       490
     emotion       0.83      0.82      0.82        82
   lifestyle       0.90      0.90      0.90       205
    personal       0.89      0.89      0.89       347
     sadness       0.88      0.91      0.90       376

    accuracy                           0.88      1500
   macro avg       0.87      0.87      0.87      1500
weighted avg       0.88      0.88      0.88      1500

Evaluating BNB max_features = 200
Confusion Matrix:
 [[378  12  23  47  30]
 [ 13  35  13   9  12]
 [ 28   3 141  13  20]
 [ 61   9  20 235  22]
 [ 53  11  33  24 255]]
Classification Report:
               precision    recall  f1-score   support

        dark       0.71      0.77      0.74       490
     emotion       0.50      0.43      0.46        82
   lifestyle       0.61      0.69      0.65       205
    personal       0.72      0.68      0.70       347
     sadness       0.75      0.68      0.71       376

    accuracy                           0.70      1500
   macro avg       0.66      0.65      0.65      1500
weighted avg       0.70      0.70      0.70      1500

Evaluating MNB max_features = 300
Confusion Matrix:
 [[446   4   7  14  19]
 [  2  74   2   2   2]
 [  8   2 187   4   4]
 [ 11   1   4 329   2]
 [ 15   2   4   5 350]]
Classification Report:
               precision    recall  f1-score   support

        dark       0.93      0.91      0.92       490
     emotion       0.89      0.90      0.90        82
   lifestyle       0.92      0.91      0.91       205
    personal       0.93      0.95      0.94       347
     sadness       0.93      0.93      0.93       376

    accuracy                           0.92      1500
   macro avg       0.92      0.92      0.92      1500
weighted avg       0.92      0.92      0.92      1500
```

```
Evaluating BNB max_features = 300
Confusion Matrix:
 [[398   9  20  33  30]
 [  9  45  13   4  11]
 [ 22   4 155  10  14]
 [ 48   8  16 257  18]
 [ 43  11  29  18 275]]
Classification Report:
               precision    recall  f1-score   support

        dark       0.77      0.81      0.79       490
     emotion       0.58      0.55      0.57        82
   lifestyle       0.67      0.76      0.71       205
    personal       0.80      0.74      0.77       347
     sadness       0.79      0.73      0.76       376

    accuracy                           0.75      1500
   macro avg       0.72      0.72      0.72      1500
weighted avg       0.76      0.75      0.75      1500

Evaluating MNB max_features = 400
Confusion Matrix:
 [[463   3   4   7  13]
 [  2  75   2   2   1]
 [  7   1 190   3   4]
 [  7   1   4 333   2]
 [ 12   1   3   4 356]]
Classification Report:
               precision    recall  f1-score   support

        dark       0.94      0.94      0.94       490
     emotion       0.93      0.91      0.92        82
   lifestyle       0.94      0.93      0.93       205
    personal       0.95      0.96      0.96       347
     sadness       0.95      0.95      0.95       376

    accuracy                           0.94      1500
   macro avg       0.94      0.94      0.94      1500
weighted avg       0.94      0.94      0.94      1500

Evaluating BNB max_features = 400
Confusion Matrix:
 [[411   9  16  31  23]
 [  8  48  12   5   9]
 [ 21   2 157   9  16]
 [ 43   5  15 267  17]
 [ 39   5  19  23 290]]
Classification Report:
               precision    recall  f1-score   support

        dark       0.79      0.84      0.81       490
     emotion       0.70      0.59      0.64        82
   lifestyle       0.72      0.77      0.74       205
    personal       0.80      0.77      0.78       347
     sadness       0.82      0.77      0.79       376

    accuracy                           0.78      1500
   macro avg       0.76      0.75      0.75      1500
weighted avg       0.78      0.78      0.78      1500
```

```
Evaluating MNB max_features = 500
Confusion Matrix:
 [[466   3   4   6  11]
 [  2  75   2   2   1]
 [  7   0 191   3   4]
 [  6   1   4 332   4]
 [  8   1   4   6 357]]
Classification Report:
               precision    recall  f1-score   support

        dark       0.95      0.95      0.95       490
     emotion       0.94      0.91      0.93        82
   lifestyle       0.93      0.93      0.93       205
    personal       0.95      0.96      0.95       347
     sadness       0.95      0.95      0.95       376

    accuracy                           0.95      1500
   macro avg       0.94      0.94      0.94      1500
weighted avg       0.95      0.95      0.95      1500

Evaluating BNB max_features = 500
Confusion Matrix:
 [[410   9  18  33  20]
 [  5  51  14   5   7]
 [ 16   2 164  10  13]
 [ 43   3  13 267  21]
 [ 29   6  20  19 302]]
Classification Report:
               precision    recall  f1-score   support

        dark       0.82      0.84      0.83       490
     emotion       0.72      0.62      0.67        82
   lifestyle       0.72      0.80      0.76       205
    personal       0.80      0.77      0.78       347
     sadness       0.83      0.80      0.82       376

    accuracy                           0.80      1500
   macro avg       0.78      0.77      0.77      1500
weighted avg       0.80      0.80      0.80      1500

Evaluating MNB max_features = 1000
Confusion Matrix:
 [[473   1   3   4   9]
 [  0  79   1   1   1]
 [  5   0 195   1   4]
 [  7   1   2 335   2]
 [  9   1   2   4 360]]
Classification Report:
               precision    recall  f1-score   support

        dark       0.96      0.97      0.96       490
     emotion       0.96      0.96      0.96        82
   lifestyle       0.96      0.95      0.96       205
    personal       0.97      0.97      0.97       347
     sadness       0.96      0.96      0.96       376

    accuracy                           0.96      1500
   macro avg       0.96      0.96      0.96      1500
weighted avg       0.96      0.96      0.96      1500
```

```
Evaluating BNB max_features = 1000
Confusion Matrix:
 [[429   1  14  27  19]
 [  4  43   9  10  16]
 [ 11   1 169   9  15]
 [ 30   2   8 282  25]
 [ 25   2  10  11 328]]
Classification Report:
               precision    recall  f1-score   support

        dark       0.86      0.88      0.87       490
     emotion       0.88      0.52      0.66        82
   lifestyle       0.80      0.82      0.81       205
    personal       0.83      0.81      0.82       347
     sadness       0.81      0.87      0.84       376

    accuracy                           0.83      1500
   macro avg       0.84      0.78      0.80      1500
weighted avg       0.84      0.83      0.83      1500

Evaluating MNB max_features = 2000
Confusion Matrix:
 [[476   1   3   4   6]
 [  0  79   1   1   1]
 [  1   0 198   2   4]
 [  3   1   2 338   3]
 [  6   0   2   4 364]]
Classification Report:
               precision    recall  f1-score   support

        dark       0.98      0.97      0.98       490
     emotion       0.98      0.96      0.97        82
   lifestyle       0.96      0.97      0.96       205
    personal       0.97      0.97      0.97       347
     sadness       0.96      0.97      0.97       376

    accuracy                           0.97      1500
   macro avg       0.97      0.97      0.97      1500
weighted avg       0.97      0.97      0.97      1500

Evaluating BNB max_features = 2000
Confusion Matrix:
 [[438   1  12  16  23]
 [  5  31   9  11  26]
 [ 11   0 157   5  32]
 [ 17   0   4 296  30]
 [ 10   0   2   3 361]]
Classification Report:
               precision    recall  f1-score   support

        dark       0.91      0.89      0.90       490
     emotion       0.97      0.38      0.54        82
   lifestyle       0.85      0.77      0.81       205
    personal       0.89      0.85      0.87       347
     sadness       0.76      0.96      0.85       376

    accuracy                           0.86      1500
   macro avg       0.88      0.77      0.80      1500
weighted avg       0.87      0.86      0.85      1500
```

```
Evaluating MNB max_features = 3000
Confusion Matrix:
 [[477   1   2   6   4]
 [  0  78   1   1   2]
 [  1   0 200   1   3]
 [  2   1   1 339   4]
 [  1   0   2   4 369]]
Classification Report:
               precision    recall  f1-score   support

         dark       0.99      0.97      0.98       490
      emotion       0.97      0.95      0.96        82
    lifestyle       0.97      0.98      0.97       205
     personal       0.97      0.98      0.97       347
      sadness       0.97      0.98      0.97       376

     accuracy                           0.98      1500
    macro avg       0.97      0.97      0.97      1500
 weighted avg       0.98      0.98      0.98      1500

Evaluating BNB max_features = 3000
Confusion Matrix:
 [[455   0   1  11  23]
 [ 10  15   7  15  35]
 [ 10   0 144   7  44]
 [ 15   0   2 292  38]
 [  6   0   1   2 367]]
Classification Report:
               precision    recall  f1-score   support

         dark       0.92      0.93      0.92       490
      emotion       1.00      0.18      0.31        82
    lifestyle       0.93      0.70      0.80       205
     personal       0.89      0.84      0.87       347
      sadness       0.72      0.98      0.83       376

     accuracy                           0.85      1500
    macro avg       0.89      0.73      0.75      1500
 weighted avg       0.87      0.85      0.84      1500

Evaluating MNB max_features = 5000
Confusion Matrix:
 [[481   0   1   4   4]
 [  0  78   1   0   3]
 [  1   0 200   1   3]
 [  2   0   1 340   4]
 [  0   0   1   4 371]]
Classification Report:
               precision    recall  f1-score   support

         dark       0.99      0.98      0.99       490
      emotion       1.00      0.95      0.97        82
    lifestyle       0.98      0.98      0.98       205
     personal       0.97      0.98      0.98       347
      sadness       0.96      0.99      0.98       376

     accuracy                           0.98      1500
    macro avg       0.98      0.97      0.98      1500
 weighted avg       0.98      0.98      0.98      1500
```

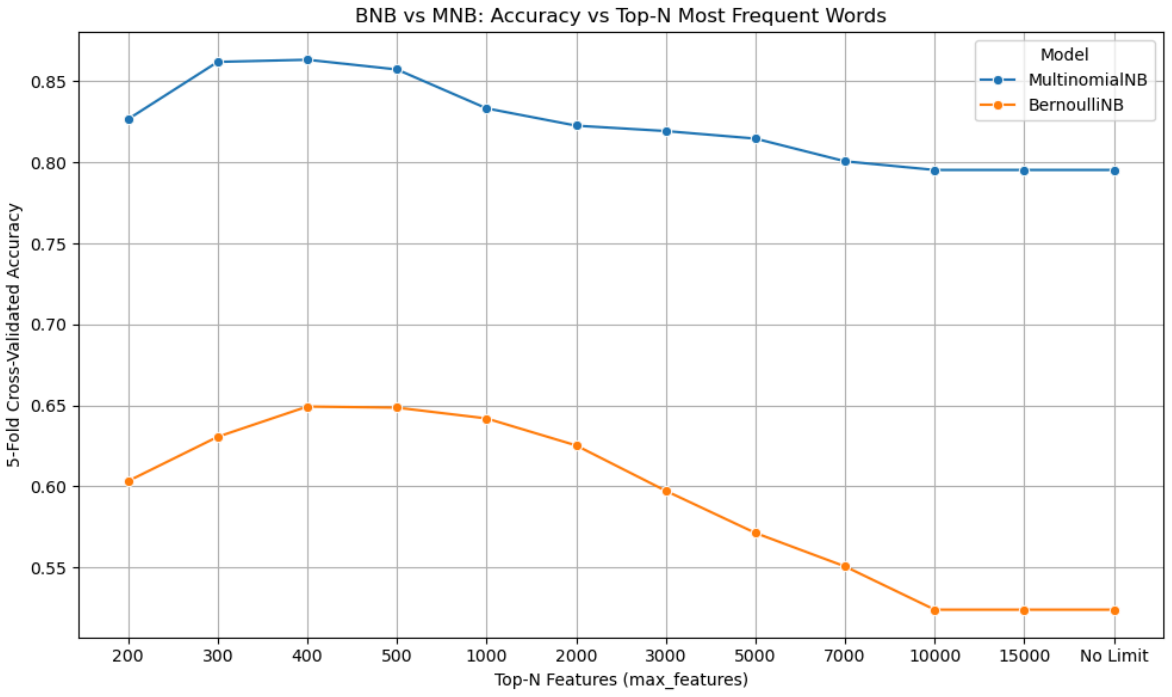
```
Evaluating BNB max_features = 5000
Confusion Matrix:
 [[456   0   0   5  29]
 [ 13   7   4  12  46]
 [ 23   0 106   5  71]
 [ 13   0   0 292  42]
 [  2   0   1   1 372]]
Classification Report:
               precision    recall  f1-score   support

        dark       0.90      0.93      0.91       490
     emotion       1.00      0.09      0.16        82
   lifestyle       0.95      0.52      0.67       205
    personal       0.93      0.84      0.88       347
     sadness       0.66      0.99      0.79       376

    accuracy                           0.82      1500
   macro avg       0.89      0.67      0.68      1500
weighted avg       0.86      0.82      0.80      1500

Evaluating MNB max_features = 7000
Confusion Matrix:
 [[482   0   0   4   4]
 [  0  75   1   1   5]
 [  1   0 199   1   4]
 [  2   0   1 340   4]
 [  0   0   1   3 372]]
Classification Report:
               precision    recall  f1-score   support

        dark       0.99      0.98      0.99       490
     emotion       1.00      0.91      0.96        82
   lifestyle       0.99      0.97      0.98       205
    personal       0.97      0.98      0.98       347
     sadness       0.96      0.99      0.97       376

    accuracy                           0.98      1500
   macro avg       0.98      0.97      0.97      1500
weighted avg       0.98      0.98      0.98      1500

Evaluating BNB max_features = 7000
Confusion Matrix:
 [[463   1   0   4  22]
 [ 14   1   1  11  55]
 [ 28   0  72   6  99]
 [ 16   0   0 281  50]
 [  2   0   0   0 374]]
Classification Report:
               precision    recall  f1-score   support

        dark       0.89      0.94      0.91       490
     emotion       0.50      0.01      0.02        82
   lifestyle       0.99      0.35      0.52       205
    personal       0.93      0.81      0.87       347
     sadness       0.62      0.99      0.77       376

    accuracy                           0.79      1500
   macro avg       0.79      0.62      0.62      1500
weighted avg       0.82      0.79      0.76      1500
```

```
Evaluating MNB max_features = 10000
Confusion Matrix:
 [[483   0   0   3   4]
 [  0  71   1   1   9]
 [  1   0 198   1   5]
 [  1   0   1 341   4]
 [  0   0   1   1 374]]
Classification Report:
               precision    recall  f1-score   support

        dark       1.00      0.99      0.99       490
     emotion       1.00      0.87      0.93        82
   lifestyle       0.99      0.97      0.98       205
    personal       0.98      0.98      0.98       347
     sadness       0.94      0.99      0.97       376

    accuracy                           0.98      1500
   macro avg       0.98      0.96      0.97      1500
weighted avg       0.98      0.98      0.98      1500

Evaluating BNB max_features = 10000
Confusion Matrix:
 [[466   2   0   2  20]
 [ 13   1   0  11  57]
 [ 30   0  56   7 112]
 [ 14   0   0 278  55]
 [  0   0   0   0 376]]
Classification Report:
               precision    recall  f1-score   support

        dark       0.89      0.95      0.92       490
     emotion       0.33      0.01      0.02        82
   lifestyle       1.00      0.27      0.43       205
    personal       0.93      0.80      0.86       347
     sadness       0.61      1.00      0.76       376

    accuracy                           0.78      1500
   macro avg       0.75      0.61      0.60      1500
weighted avg       0.81      0.78      0.75      1500

Evaluating MNB max_features = 15000
Confusion Matrix:
 [[483   0   0   3   4]
 [  0  71   1   1   9]
 [  1   0 198   1   5]
 [  1   0   1 341   4]
 [  0   0   1   1 374]]
Classification Report:
               precision    recall  f1-score   support

        dark       1.00      0.99      0.99       490
     emotion       1.00      0.87      0.93        82
   lifestyle       0.99      0.97      0.98       205
    personal       0.98      0.98      0.98       347
     sadness       0.94      0.99      0.97       376

    accuracy                           0.98      1500
   macro avg       0.98      0.96      0.97      1500
weighted avg       0.98      0.98      0.98      1500
```

```
Evaluating BNB max_features = 15000
Confusion Matrix:
 [[466   2   0   2  20]
 [ 13   1   0  11  57]
 [ 30   0  56   7 112]
 [ 14   0   0 278  55]
 [  0   0   0   0 376]]
Classification Report:
               precision    recall  f1-score   support

        dark       0.89      0.95      0.92       490
     emotion       0.33      0.01      0.02        82
   lifestyle       1.00      0.27      0.43       205
    personal       0.93      0.80      0.86       347
     sadness       0.61      1.00      0.76       376

    accuracy                           0.78      1500
   macro avg       0.75      0.61      0.60      1500
weighted avg       0.81      0.78      0.75      1500

Evaluating MNB max_features = None
Confusion Matrix:
 [[483   0   0   3   4]
 [  0  71   1   1   9]
 [  1   0 198   1   5]
 [  1   0   1 341   4]
 [  0   0   1   1 374]]
Classification Report:
               precision    recall  f1-score   support

        dark       1.00      0.99      0.99       490
     emotion       1.00      0.87      0.93        82
   lifestyle       0.99      0.97      0.98       205
    personal       0.98      0.98      0.98       347
     sadness       0.94      0.99      0.97       376

    accuracy                           0.98      1500
   macro avg       0.98      0.96      0.97      1500
weighted avg       0.98      0.98      0.98      1500

Evaluating BNB max_features = None
Confusion Matrix:
 [[466   2   0   2  20]
 [ 13   1   0  11  57]
 [ 30   0  56   7 112]
 [ 14   0   0 278  55]
 [  0   0   0   0 376]]
Classification Report:
               precision    recall  f1-score   support

        dark       0.89      0.95      0.92       490
     emotion       0.33      0.01      0.02        82
   lifestyle       1.00      0.27      0.43       205
    personal       0.93      0.80      0.86       347
     sadness       0.61      1.00      0.76       376

    accuracy                           0.78      1500
   macro avg       0.75      0.61      0.60      1500
weighted avg       0.81      0.78      0.75      1500
```

Result chart + Best N description

In [10]:
```
import matplotlib.pyplot as plt
import seaborn as sns

results_df_Mnb["Model"] = "MultinomialNB"
results_df_Bnb["Model"] = "BernoulliNB"
plot_df = pd.concat([results_df_Mnb, results_df_Bnb], ignore_index=True)

plot_df["Top-N Features"] = plot_df["Top-N Features"].astype(str)

plt.figure(figsize=(10, 6))
sns.lineplot(data=plot_df, x="Top-N Features", y="Accuracy", hue="Model", marker
plt.title("BNB vs MNB: Accuracy vs Top-N Most Frequent Words")
plt.xlabel("Top-N Features (max_features)")
plt.ylabel("5-Fold Cross-Validated Accuracy")
plt.grid(True)
plt.tight_layout()
plt.show()

print("MNB Accuracy Results:")
print(results_df_Mnb.sort_values(by="Accuracy", ascending=False))
print("\nBNB Accuracy Results:")
print(results_df_Bnb.sort_values(by="Accuracy", ascending=False))
```



```
MNB Accuracy Results:
   Top-N Features  Accuracy          Model
2             400  0.863333  MultinomialNB
1             300  0.862000  MultinomialNB
3             500  0.857333  MultinomialNB
4            1000  0.833333  MultinomialNB
0             200  0.826667  MultinomialNB
5            2000  0.822667  MultinomialNB
6            3000  0.819333  MultinomialNB
7            5000  0.814667  MultinomialNB
8            7000  0.800667  MultinomialNB
9           10000  0.795333  MultinomialNB
10          15000  0.795333  MultinomialNB
11       No Limit  0.795333  MultinomialNB

BNB Accuracy Results:
   Top-N Features  Accuracy        Model
2             400  0.649333  BernoulliNB
3             500  0.648667  BernoulliNB
4            1000  0.642000  BernoulliNB
1             300  0.630667  BernoulliNB
5            2000  0.625333  BernoulliNB
0             200  0.603333  BernoulliNB
6            3000  0.597333  BernoulliNB
7            5000  0.571333  BernoulliNB
8            7000  0.550667  BernoulliNB
9           10000  0.524000  BernoulliNB
10          15000  0.524000  BernoulliNB
11       No Limit  0.524000  BernoulliNB
```

Top-N feature number optimal selection and experimental results

The control of the number of most common words preserved in the process of text vectorization is obtained through `CountVectorizer(max_features=N)`, and 50% cross-validation on MultinomialNB (MNB) and BernoulliNB (BNB) is conducted respectively under different N values. The result looks like above.

Experimental results and conclusions:

MNB outperforms BNB across all N values and achieves the highest rate of accuracy, i.e., 86.3%, when Top-N = 400.

The performance of BNB was overall very bad. It performed best (64.9%) when N = 400, but was significantly worse than MNB.

The final selection

Top-N Features =400, as a constant parameter to `CountVectorizer(max_features=400)` used in subsequent operations.

Part 1 Q5

Logistic Regression Method

Logistic regression is a linear classifier with extensive applications in multi-classification problems, and its output is the probability of a class. In contrast to Naive Bayes's conditional independent hypothesis, logistic regression has the ability to model the linear correlation between classes and features by learning the weights, and typically achieves incredible performance in tasks such as text classification and advertisement click-through rate prediction.

The reasons why logistic regression is chosen for this data set are:

- The feature dimension is relatively high and sparse once the lyrics are vectorized by the word bag;
- Logistic regression can remain effective in high-dimensional sparse feature spaces;
- It is not sensitive to collinearity among features and can accept word frequency vector input.

Therefore, I choose to utilize logistic regression as the baseline model.

Model import and data vectorization

In [11]:
```
from sklearn.linear_model import LogisticRegression

vectorizer_count = CountVectorizer(max_features=400)

X_lr = vectorizer_count.fit_transform(df["clean_text"])
y = df["topic"]
```

The use of the logistic regression model is achieved through `from sklearn.linear_model import LogisticRegression`.

Model tuning

In [12]:
```
C_values = [0.01, 0.1, 0.5, 1, 2, 5, 10, 20]
results = []
for c in C_values:
    print(f"Evaluating Logistic Regression with C={c}")
    model = LogisticRegression(C=c, solver='liblinear', max_iter=1000, class_wei
    scores = cross_validate(model, X_lr, y, cv=5, scoring=scoring)
    results.append({
        "C": c,
        "Accuracy": np.mean(scores['test_accuracy']),
        "Precision": np.mean(scores['test_precision_macro']),
        "Recall": np.mean(scores['test_recall_macro']),
        "F1": np.mean(scores['test_f1_macro'])
    })

results_df = pd.DataFrame(results)
print(results_df.sort_values(by="F1", ascending=False))
```

```
Evaluating Logistic Regression with C=0.01
Evaluating Logistic Regression with C=0.1
Evaluating Logistic Regression with C=0.5
Evaluating Logistic Regression with C=1
Evaluating Logistic Regression with C=2
Evaluating Logistic Regression with C=5
Evaluating Logistic Regression with C=10
Evaluating Logistic Regression with C=20
       C  Accuracy  Precision    Recall        F1
1   0.10  0.883333   0.871378  0.866155  0.867820
0   0.01  0.877333   0.861756  0.866226  0.863328
2   0.50  0.878000   0.866195  0.856960  0.860877
3   1.00  0.874000   0.857401  0.851430  0.853584
4   2.00  0.870000   0.852938  0.843346  0.847255
5   5.00  0.868000   0.848887  0.842481  0.844856
6  10.00  0.861333   0.843689  0.834542  0.837652
7  20.00  0.857333   0.838970  0.828705  0.832321
```

Due to the limited dataset and sparsity of features, parameters are given as follows:

- **solver='liblinear'** : Suitable for small datasets and can deal with L2 regularization;
- **max_iter=1000** : Ensure that iterations are sufficiently large so that the model converges.
- **C=0.1** : Strong regularization strength with a lower value of C is taken in this case to control model complexity and avoid overfitting for high-dimensional sparse text representations. This setting led to the best Macro-F1 performance during cross-validation and thus was selected as the final model parameter.

Hypothesis: Compared to the Naive Bayes model (BNB and MNB), **Logistic Regression may have a higher macro average F1 score** in lyrics topic classification task, as it doesn't assume feature independence and can better model the conconnectives in context, being suitable for sparse high-dimensional feature Spaces.

Lastly, I will use cross-validation and the same preprocessing method to compare the accuracy and macro average F1 score of the three models to determine whether this hypothesis holds true.

Set cross-validation scoring metrics and conduct evaluations

In [13]:
```
lr_model = LogisticRegression(max_iter=1000, solver='liblinear', C=0.1, class_we
lr_scores = cross_validate(lr_model, X_lr, y, cv=5, scoring=scoring)

mnb_acc = evaluate_with_metrics(df["clean_text"], df["topic"], max_features=400,
mnb_precesion = evaluate_with_metrics(df["clean_text"], df["topic"], max_feature
mnb_recall = evaluate_with_metrics(df["clean_text"], df["topic"], max_features=4
mnb_f1 = evaluate_with_metrics(df["clean_text"], df["topic"], max_features=400,

bnb_acc = evaluate_with_metrics(df["clean_text"], df["topic"], max_features=400,
bnb_precesion = evaluate_with_metrics(df["clean_text"], df["topic"], max_feature
bnb_recall = evaluate_with_metrics(df["clean_text"], df["topic"], max_features=4
bnb_f1 = evaluate_with_metrics(df["clean_text"], df["topic"], max_features=400,
```

```
Confusion Matrix:
 [[463   3   4   7  13]
 [  2  75   2   2   1]
 [  7   1 190   3   4]
 [  7   1   4 333   2]
 [ 12   1   3   4 356]]
Classification Report:
               precision    recall  f1-score   support

        dark       0.94      0.94      0.94       490
     emotion       0.93      0.91      0.92        82
   lifestyle       0.94      0.93      0.93       205
    personal       0.95      0.96      0.96       347
     sadness       0.95      0.95      0.95       376

    accuracy                           0.94      1500
   macro avg       0.94      0.94      0.94      1500
weighted avg       0.94      0.94      0.94      1500

Confusion Matrix:
 [[463   3   4   7  13]
 [  2  75   2   2   1]
 [  7   1 190   3   4]
 [  7   1   4 333   2]
 [ 12   1   3   4 356]]
Classification Report:
               precision    recall  f1-score   support

        dark       0.94      0.94      0.94       490
     emotion       0.93      0.91      0.92        82
   lifestyle       0.94      0.93      0.93       205
    personal       0.95      0.96      0.96       347
     sadness       0.95      0.95      0.95       376

    accuracy                           0.94      1500
   macro avg       0.94      0.94      0.94      1500
weighted avg       0.94      0.94      0.94      1500

Confusion Matrix:
 [[463   3   4   7  13]
 [  2  75   2   2   1]
 [  7   1 190   3   4]
 [  7   1   4 333   2]
 [ 12   1   3   4 356]]
Classification Report:
               precision    recall  f1-score   support

        dark       0.94      0.94      0.94       490
     emotion       0.93      0.91      0.92        82
   lifestyle       0.94      0.93      0.93       205
    personal       0.95      0.96      0.96       347
     sadness       0.95      0.95      0.95       376

    accuracy                           0.94      1500
   macro avg       0.94      0.94      0.94      1500
weighted avg       0.94      0.94      0.94      1500

Confusion Matrix:
 [[463   3   4   7  13]
 [  2  75   2   2   1]
```

```
 [  7   1 190   3   4]
 [  7   1   4 333   2]
 [ 12   1   3   4 356]]
Classification Report:
              precision    recall  f1-score   support

        dark       0.94      0.94      0.94       490
     emotion       0.93      0.91      0.92        82
   lifestyle       0.94      0.93      0.93       205
    personal       0.95      0.96      0.96       347
     sadness       0.95      0.95      0.95       376

    accuracy                           0.94      1500
   macro avg       0.94      0.94      0.94      1500
weighted avg       0.94      0.94      0.94      1500

Confusion Matrix:
 [[411   9  16  31  23]
 [  8  48  12   5   9]
 [ 21   2 157   9  16]
 [ 43   5  15 267  17]
 [ 39   5  19  23 290]]
Classification Report:
              precision    recall  f1-score   support

        dark       0.79      0.84      0.81       490
     emotion       0.70      0.59      0.64        82
   lifestyle       0.72      0.77      0.74       205
    personal       0.80      0.77      0.78       347
     sadness       0.82      0.77      0.79       376

    accuracy                           0.78      1500
   macro avg       0.76      0.75      0.75      1500
weighted avg       0.78      0.78      0.78      1500

Confusion Matrix:
 [[411   9  16  31  23]
 [  8  48  12   5   9]
 [ 21   2 157   9  16]
 [ 43   5  15 267  17]
 [ 39   5  19  23 290]]
Classification Report:
              precision    recall  f1-score   support

        dark       0.79      0.84      0.81       490
     emotion       0.70      0.59      0.64        82
   lifestyle       0.72      0.77      0.74       205
    personal       0.80      0.77      0.78       347
     sadness       0.82      0.77      0.79       376

    accuracy                           0.78      1500
   macro avg       0.76      0.75      0.75      1500
weighted avg       0.78      0.78      0.78      1500

Confusion Matrix:
 [[411   9  16  31  23]
 [  8  48  12   5   9]
 [ 21   2 157   9  16]
 [ 43   5  15 267  17]
 [ 39   5  19  23 290]]
```

```
Classification Report:
              precision    recall  f1-score   support

        dark       0.79      0.84      0.81       490
     emotion       0.70      0.59      0.64        82
   lifestyle       0.72      0.77      0.74       205
    personal       0.80      0.77      0.78       347
     sadness       0.82      0.77      0.79       376

    accuracy                           0.78      1500
   macro avg       0.76      0.75      0.75      1500
weighted avg       0.78      0.78      0.78      1500

Confusion Matrix:
 [[411   9  16  31  23]
 [  8  48  12   5   9]
 [ 21   2 157   9  16]
 [ 43   5  15 267  17]
 [ 39   5  19  23 290]]
Classification Report:
              precision    recall  f1-score   support

        dark       0.79      0.84      0.81       490
     emotion       0.70      0.59      0.64        82
   lifestyle       0.72      0.77      0.74       205
    personal       0.80      0.77      0.78       347
     sadness       0.82      0.77      0.79       376

    accuracy                           0.78      1500
   macro avg       0.76      0.75      0.75      1500
weighted avg       0.78      0.78      0.78      1500
```

The results were summarized and compared with BNB/MNB

In [14]:
```
final_comparison = pd.DataFrame({
    "Metric": ['Accuracy', 'Macro F1'],
    "Logistic Regression": [
        np.mean(lr_scores['test_accuracy']),
        np.mean(lr_scores['test_f1_macro'])
    ],
    "MultinomialNB": [mnb_acc, mnb_f1],
    "BernoulliNB": [bnb_acc, bnb_f1]
})

final_comparison
```

Out[14]:

	Metric	Logistic Regression	MultinomialNB	BernoulliNB
0	Accuracy	0.883333	0.863333	0.649333
1	Macro F1	0.867820	0.844955	0.564765

Hypothesis Verification

According to the comparison of final score, it is found that compared to BNB and MNB, **logistic regression can perform better on Macro-F1 score and Accuracy, especially**

with stronger generalization power in the high-dimensional sparse feature space. The assumption holds. A higher Macro-F1 score implies that it maintains a more balanced performance among various categories (including subcategories). The Accuracy also gets the best value, implying that its overall prediction performance is the best.

Final model configuration

- Preprocessing text: **lower + nltk + stemming + punkt tokenizer**
- Features to set: **max_features=400**
- Classifier to use: **Logistic Regression**
- Strength of regularization: **C = 0.1**

It does its best or less than optimally on all evaluation criteria. It possesses good interpretability, computational efficiency and generalization capability, and is the "overall best" topic classification model solution for this task.

Part 2

Part2 Q1

Load data & Build the training test set

In [15]:
```
train_data = df.iloc[:750].copy()
train_data["document"] = train_data["artist_name"].fillna("") + " " + \
                 train_data["track_name"].fillna("") + " " + \
                 train_data["genre"].fillna("") + " " + \
                 train_data["lyrics"].fillna("")
train_data.reset_index(drop=True, inplace=True)
train_data["clean_lyrics"] = train_data["lyrics"].apply(lambda x: preprocess(x,
train_data["clean_text"] = train_data["document"].apply(lambda x: preprocess(x,


test_data = df.iloc[750:1000].copy()
test_data["document"] = test_data["artist_name"].fillna("") + " " + \
                 test_data["track_name"].fillna("") + " " + \
                 test_data["genre"].fillna("") + " " + \
                 test_data["lyrics"].fillna("")
test_data.reset_index(drop=True, inplace=True)
test_data["clean_lyrics"] = test_data["lyrics"].apply(lambda x: preprocess(x, Tr
test_data["clean_text"] = test_data["document"].apply(lambda x: preprocess(x, Tr
```

Save vectorizer and model for Part2

In [16]:
```
from joblib import dump
from joblib import load
from sklearn.feature_extraction.text import TfidfVectorizer

nltk_stop_words_list = stopwords.words('english')

vectorizer_tfidf = TfidfVectorizer(max_features = 400)
```

```
vectorizer_tfidf.fit(train_data["clean_text"])
dump(vectorizer_tfidf, 'best_vectorizer_tfidf.joblib')
best_vectorizer_tfidf = load('best_vectorizer_tfidf.joblib')

X_lr = vectorizer_tfidf.fit_transform(train_data["clean_text"])
y = train_data["topic"]
lr_model.fit(X_lr, y)
dump(lr_model, 'best_model.joblib')
best_clf = load('best_model.joblib')
```

Use the best classifier in Part 1 to predict the topic tags of the training set

In [17]:
```
X_train_vec = best_vectorizer_tfidf.transform(train_data["clean_text"])
train_data.loc[:, "predicted_topic"] = best_clf.predict(X_train_vec)
```

Read the user's keyword file and find out the songs in the training set that each user likes

In [18]:
```
def preprocess_keywords(keywords):
    return set(preprocess(' '.join(keywords), True, False, True, True).split())

def load_user_keywords(filepath):
    with open(filepath, 'r') as f:
        lines = f.readlines()
    topic_keywords = {}
    for line in lines:
        topic, keywords = line.strip().split('\t')
        topic = topic.strip().lower()
        keyword_list = re.findall(r'\w+', keywords.lower())
        keyword_list = preprocess_keywords(keyword_list)
        topic_keywords[topic] = keyword_list
    return topic_keywords


user1_keywords = load_user_keywords('user1.tsv')
user2_keywords = load_user_keywords('user2.tsv')

# Determine whether a user likes a certain song
def liked_songs(user_keywords, df):
    liked = {topic: [] for topic in user_keywords}
    for _, row in df.iterrows():
        topic = row['predicted_topic']
        lyric = row['clean_lyrics'].lower()
        if topic in user_keywords:
            if any(kw in lyric for kw in user_keywords[topic]):
                liked[topic].append(lyric)
    return liked


user1_liked = liked_songs(user1_keywords, train_data)
user2_liked = liked_songs(user2_keywords, train_data)
```

Build user profiles

In [19]:
```
def build_user_profile(liked_dict, vec):
    corpus = []
    for topic in ['dark', 'emotion', 'lifestyle', 'personal', 'sadness']:
        topic_text = ' '.join(liked_dict.get(topic, []))
        corpus.append(topic_text)
    tfidf_matrix = vec.transform(corpus)
    return tfidf_matrix

user1_profile = build_user_profile(user1_liked, best_vectorizer_tfidf)
user2_profile = build_user_profile(user2_liked, best_vectorizer_tfidf)
```

Output the top 20 keywords for each topic in the user profile

In [20]:
```
def print_top_keywords(tfidf_matrix, vec, user_name):
    print(f"\n===== {user_name} Top 20 keywords for user profiling=====\n")
    topics = ['dark', 'emotion', 'lifestyle', 'personal', 'sadness']
    for idx, topic in enumerate(topics):
        vector = tfidf_matrix[idx].toarray().flatten()
        top_n = np.argsort(vector)[::-1][:20]
        keywords = [vec.get_feature_names_out()[j] for j in top_n if vector[j] >
        print(f"{topic.capitalize()}: {keywords}\n")

print_top_keywords(user1_profile, best_vectorizer_tfidf, 'User 1')
print_top_keywords(user2_profile, best_vectorizer_tfidf, 'User 2')
```

```
===== User 1 Top 20 keywords for user profiling=====

Dark: ['fight', 'blood', 'grind', 'like', 'na', 'stand', 'gon', 'know', 'kill',
'come', 'hear', 'beat', 'black', 'hand', 'tell', 'teach', 'rise', 'yeah', 'buil
d', 'pain']

Emotion: ['good', 'touch', 'feel', 'hold', 'feelin', 'year', 'kiss', 'vibe', 'mor
n', 'miss', 'want', 'luck', 'go', 'lovin', 'lip', 'sunris', 'know', 'light', 'cau
s', 'na']

Lifestyle: ['tonight', 'song', 'night', 'sing', 'home', 'come', 'stranger', 'lon
g', 'time', 'right', 'wait', 'spoil', 'tire', 'na', 'wan', 'telephon', 'struggl',
'play', 'mind', 'babi']

Personal: ['life', 'live', 'chang', 'world', 'ordinari', 'na', 'thank', 'dream',
'know', 'yeah', 'lord', 'wan', 'like', 'thing', 'time', 'think', 'learn', 'oohooh
oohooh', 'grow', 'need']

Sadness: ['tear', 'break', 'babi', 'fall', 'cri', 'heart', 'away', 'woah', 'na',
'hurt', 'know', 'wish', 'think', 'apart', 'want', 'feel', 'wan', 'place', 'leav',
'caus']


===== User 2 Top 20 keywords for user profiling=====

Dark: []

Emotion: ['good', 'touch', 'feel', 'hold', 'feelin', 'year', 'kiss', 'vibe', 'mor
n', 'miss', 'want', 'luck', 'go', 'lovin', 'lip', 'sunris', 'know', 'light', 'cau
s', 'na']

Lifestyle: []

Personal: []

Sadness: ['tear', 'heart', 'break', 'away', 'fall', 'cri', 'insid', 'na', 'woah',
'babi', 'hurt', 'step', 'know', 'open', 'apart', 'wish', 'like', 'feel', 'wan',
'think']
```

User 1 Profile Analysis

- **Dark**: Keywords such as "*fight*", "*blood*", "*kill*", and "*pain*" convey strong themes of violence and conflict, which are characteristic of darker lyrical content. This vector indicates heavy, possibly aggressive songs.
- **Emotion**: Words such as "*touch*", "*feel*", "*kiss*", "*vibe*", and "*morning*" express a strong emotional and sensual focus with a touch of sensitivity to relational or romantic content.
- **Lifestyle**: Words such as "*tonight*", "*night*", "*home*", and "*song*" imply focus on social or contemplative topics, usually associated with lifestyle and daily life.
- **Personal**: With oft-repeated words like "*life*", "*change*", "*dream*", and "*learn*", this vector represents songs that talk of self-improvement, introspection, and existential issues.
- **Sadness**: Words like "*tear*", "*break*", "*cry*", "*hurt*", and "*heart*" are strongly linked with the motifs of loss and vulnerability, suggesting deep involvement with sad content.

User 2 Profile Analysis

- **Emotion**: Shares many keywords with User 1, including "*feel*", "*kiss*", "*vibe*", and "*lovin*", confirming that this user is likely very focused on relational and sensual aspects of music.
- **Sadness**: The presence of "*tear*", "*hurt*", "*cry*", and "*heart*" indicates alignment with emotionally vulnerable and melancholic songs.

Summary

Both users' profiles appear to be consistent and interpretable:

- **User 1** has a rich and diverse emotional life, with participation in all five themes.
- **User 2** has a more selective emotional life, with a focus on romantic and sad content.

Define User3 and customize keywords to build an interest profile

In [21]:
```
user3_keywords = {
    'dark': {"night", "club", "dance", "party"},
    'emotion': {"love", "heart", "baby", "kiss"},
    'lifestyle': {"money", "dream", "city", "street"},
    'personal': {"diary", "life", "alone", "feel"},
    'sadness': {"tears", "cry", "hurt", "pain"}
}

user3_keywords = {topic: preprocess_keywords(words) for topic, words in user3_ke
user3_liked = liked_songs(user3_keywords, train_data)
user3_profile = build_user_profile(user3_liked, best_vectorizer_tfidf)
print_top_keywords(user3_profile, best_vectorizer_tfidf, 'User 3')
```

```
===== User 3 Top 20 keywords for user profiling=====

Dark: ['closer', 'fight', 'black', 'night', 'shout', 'come', 'na', 'welcom', 'fol
low', 'oooh', 'readi', 'gon', 'ring', 'stand', 'head', 'light', 'ghost', 'like',
'know', 'evil']

Emotion: ['good', 'go', 'touch', 'hold', 'feel', 'feelin', 'kiss', 'vibe', 'mor
n', 'miss', 'want', 'luck', 'lovin', 'lip', 'sunris', 'know', 'babi', 'na', 'wai
t', 'like']

Lifestyle: ['spoil', 'song', 'play', 'home', 'night', 'come', 'right', 'wait', 'w
ant', 'time', 'make', 'belong', 'root', 'know', 'summer', 'thought', 'long', 'yea
h', 'dear', 'na']

Personal: ['life', 'live', 'chang', 'world', 'ordinari', 'na', 'know', 'yeah', 'l
ord', 'give', 'wan', 'like', 'think', 'thing', 'time', 'thank', 'dream', 'oohooho
ohooh', 'grow', 'believ']

Sadness: ['break', 'hurt', 'heart', 'fall', 'tear', 'na', 'babi', 'away', 'know',
'wan', 'think', 'caus', 'cri', 'leav', 'like', 'pain', 'yeah', 'feel', 'woah', 'g
on']
```

User 3 Profiling Analysis Top 20 Keywords

- **Dark**: Keywords like **"fight"**, **"black"**, **"evil"**, **"ghost"**, and **"shout"**, indicating an interest in violent, foreboding, or rebellious lyrical subject matter.
- **Emotion**: Words such as **"feel"**, **"touch"**, **"kiss"**, **"vibe"**, **"sunrise"**, and **"miss"** are sensual and romantic in tone. Repetitive usage of the most common emotion-oriented words such as **"babi"**, **"wait"**, and **"want"** assures the emotional interest.
- **Lifestyle**: Words like **"play"**, **"night"**, **"home"**, **"song"**, and **"summer"** with a focus on habit-based, memory-based, or longing-based life.
- **Personal**: This vector is governed by reflective words: **"life"**, **"live"**, **"dream"**, **"think"**, **"grow"**, and **"believ"**, all of which refer to self-enlargement and reflection themes.
- **Sadness**: Terms such as **"break"**, **"hurt"**, **"tear"**, **"fall"**, and **"cri"** clearly refer to emotional hurt or vulnerability. Despite the sadness, repetition of terms such as **"babi"** and **"like"** show that even sad songs are romantic in nature.

Summary

User 3's profile is an intense emotional investment in music spanning **romantic**, **introspective**, **melancholic**, and **dark/edgy** themes. The keywords reflect a user who would likely be drawn to lyrics conveying deep personal states, complex emotional experiences, and maybe darker aesthetics.

Part2 Q2

Training set preprocessing

In [22]:
```
test_data["clean_text"] = test_data["document"].apply(lambda x: preprocess(x, Tr

X_test_vec_tfidf = vectorizer_tfidf.transform(test_data['clean_text'])

test_data.loc[:, 'predicted_topic'] = best_clf.predict(X_test_vec_tfidf)
```

Construction of user portraits

In [23]:
```
from sklearn.preprocessing import normalize
def build_user_profile_m(user_keywords_dict, df, vec, top_m='all'):
    profile = {}
    for topic in ['dark', 'emotion', 'lifestyle', 'personal', 'sadness']:
        liked_lyrics = []
        keywords = user_keywords_dict.get(topic, set())
        if not keywords:
            profile[topic] = vec.transform([''])
            continue
        for _, row in df[df['predicted_topic'] == topic].iterrows():
            lyric = row['clean_lyrics'].lower()
            if any(kw in lyric for kw in keywords):
                liked_lyrics.append(lyric)
        if liked_lyrics:
```

```
            text = ' '.join(liked_lyrics)
        else:
            text = ''
        vec_full = vec.transform([text])
        if top_m == 'all':
            vec_selected = vec_full
        else:
            arr = vec_full.toarray()[0]
            top_idx = arr.argsort()[::-1][:top_m]
            masked = np.zeros_like(arr)
            masked[top_idx] = arr[top_idx]
            vec_selected = vec.transform([''])
            vec_selected = vec_selected.multiply(0)
            vec_selected[0, top_idx] = masked[top_idx]
        vec_normalized = normalize(vec_selected, norm='l2')
        profile[topic] = vec_normalized
    return profile
```

Build the recommended similarity for each song in the test set

In [24]:
```
from sklearn.metrics.pairwise import cosine_similarity

def compute_similarity_scores(user_profile, song_vectors, predicted_topics):
    scores = []

    for i, topic in enumerate(predicted_topics):
        song_vec = song_vectors[i:i+1]
        if topic not in user_profile:
            scores.append(0.0)
            continue
        profile_vec = user_profile[topic]

        score = cosine_similarity(profile_vec, song_vec)[0, 0]

        scores.append(score)

    return scores
```

The top N songs recommended

In [25]:
```
def get_top_n_recommendations(df, sim_scores):
    df_copy = df.copy()
    df_copy['similarity'] = sim_scores
    df_sorted = df_copy.sort_values(by='similarity', ascending=False).head(10)
    return df_sorted.reset_index(drop=True)
```

Choosing Recommendation Size N

I set the total number of songs to recommend to **N = 10** for this experiment, based on the following considerations:

- **Cognitive Load Management**: The list is kept small so that users are not cognitively burdened with too many recommendations, reducing the quality of the feedback.

- **Realistic User Simulation**: A shorter recommendation list is more consistent with user interaction on real websites (e.g., Spotify), for which preference modeling is more realistic.
- **Evaluation Sensitivity**: A moderate value of N exposes ranking algorithm differences, making measures such as Precision and MAP more revealing.

In total, setting N = 10 is a balance of user experience, ease of evaluation, and system realism.

Evaluation metrics: Precision@N, Recall@N, F1@N, MAP

In [26]:
```
def is_song_liked(row, user_keywords):
    topic = row['predicted_topic']
    lyric = row['clean_lyrics'].lower()
    if topic in user_keywords:
        return any(kw in lyric for kw in user_keywords[topic])
    return False

def calc_f1(precision, recall):
    return 2 * precision * recall / (precision + recall + 1e-8)

def calc_map(hits):
    precisions = []
    hit_count = 0
    for i, hit in enumerate(hits):
        if hit:
            hit_count += 1
            precisions.append(hit_count / (i + 1))
    if hit_count == 0:
        return 0.0
    return sum(precisions) / hit_count

def evaluate_recommendations(recommendations, user_keywords, test_df):
    hits = [1 if is_song_liked(row, user_keywords) else 0 for _, row in recommen
    # Precision@N
    precision = sum(hits) / len(hits) if hits else 0

    # Recall@N
    test_df_copy = test_df.copy()
    test_df_copy['is_liked'] = test_df_copy.apply(lambda row: is_song_liked(row,
    total_liked_in_test = test_df_copy['is_liked'].sum()
    recall = sum(hits) / total_liked_in_test if total_liked_in_test else 0

    # F1@N
    f1 = 2 * precision * recall / (precision + recall + 1e-8) if (precision + re

    # MAP
    precisions = []
    hit_count = 0
    for i, hit in enumerate(hits):
        if hit:
            hit_count += 1
            precisions.append(hit_count / (i + 1))
    map_score = sum(precisions) / hit_count if hit_count > 0 else 0

    return precision, recall, f1, map_score
```

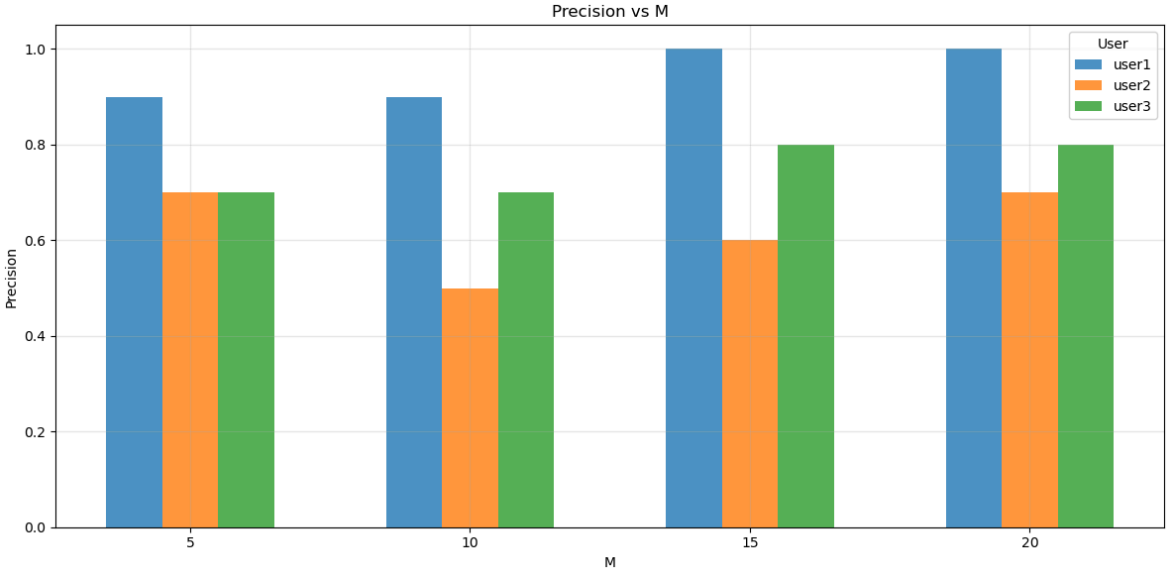
I use `Precision@N`, `Recall@N`, and `MAP` as metrics. The primary metric is Precision@N, which computes the proportion of relevant items in the recommendation list, since it is the most indicative of recommendation quality. Recall@N is complementary since it computes the proportion of the user's interest that was covered. MAP considers the rank of the relevant items and measures how effective the system is at ranking the recommendations. These metrics collectively provide a comprehensive view of accuracy, coverage, and ranking quality.

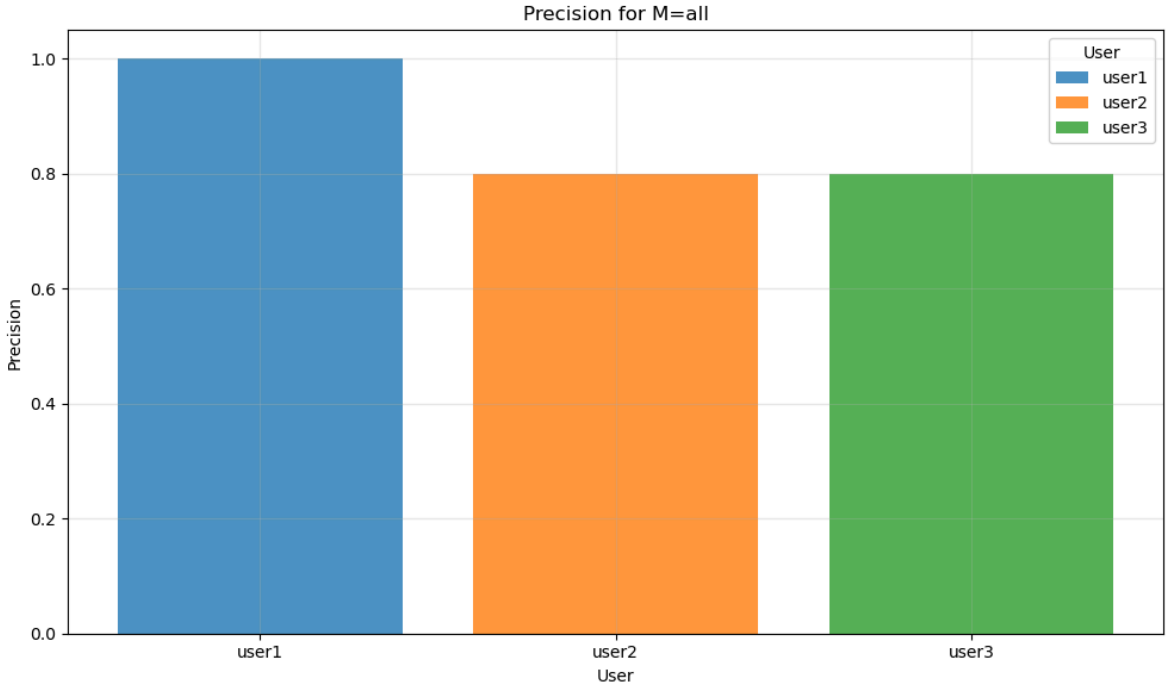
Evaluation

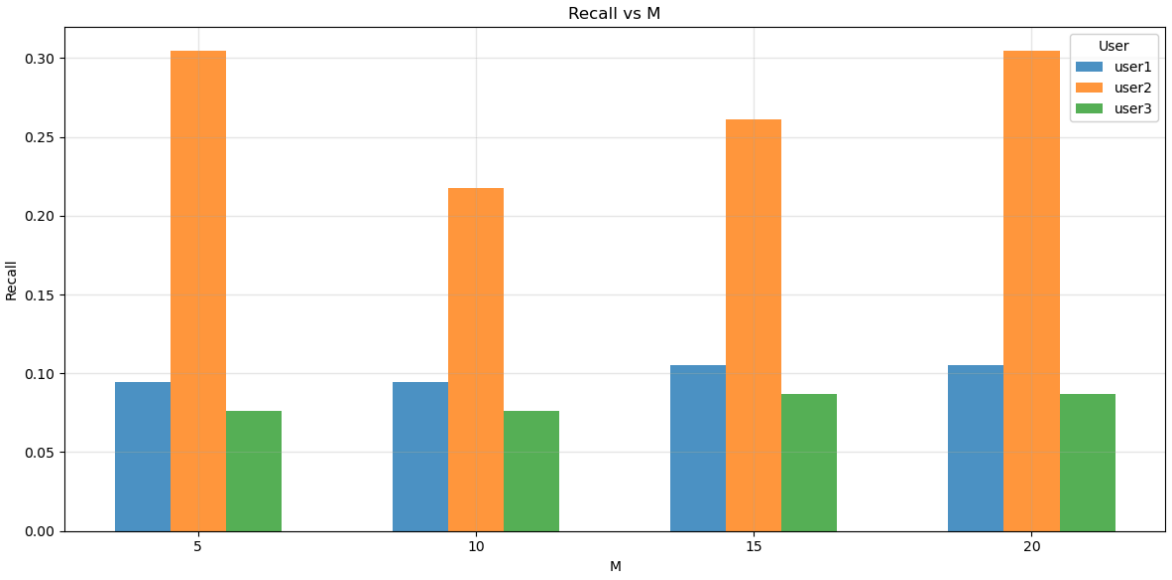
In [27]:
```
import warnings
from scipy.sparse import SparseEfficiencyWarning
warnings.filterwarnings('ignore', category=SparseEfficiencyWarning)

def final_evaluation(M_list, N, user_liked_dicts, train_data, test_data,
                                    best_vectorizer_tfidf, X_test_vec_tfidf):
    results = []
    for M in M_list:
        for uid, liked_dict in user_liked_dicts.items():

            user_profile = build_user_profile_m(liked_dict, train_data, best_vec

            sim_scores = compute_similarity_scores(user_profile, X_test_vec_tfid

            topN_df = get_top_n_recommendations(test_data, sim_scores)

            precision, recall, f1, map = evaluate_recommendations(topN_df, liked

            results.append((uid, M, precision, recall, f1, map))

    df_result = pd.DataFrame(results, columns=['User', 'M', 'Precision', 'Recall
    return df_result

M_list = [5, 10, 15, 20, 'all']
N = 10
user_liked_dicts = {
    'user1': user1_keywords,
    'user2': user2_keywords,
    'user3': user3_keywords
}

df_result = final_evaluation(
    M_list=M_list,
    N=N,
    user_liked_dicts=user_liked_dicts,
    train_data=train_data,
    test_data=test_data,
    best_vectorizer_tfidf=best_vectorizer_tfidf,
    X_test_vec_tfidf=X_test_vec_tfidf
)
```

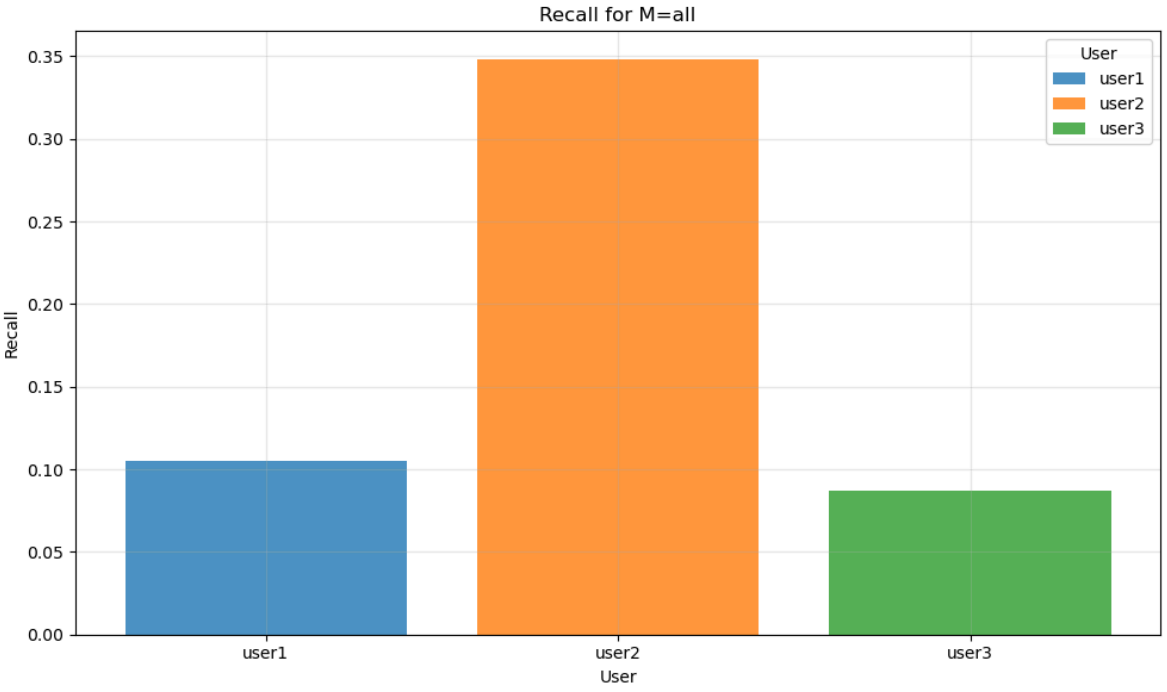
In [28]:
```
def plot_metrics_comparison(df_result):

    df_numeric = df_result[df_result['M'] != 'all'].copy()
    df_all = df_result[df_result['M'] == 'all'].copy()
    metrics = ['Precision', 'Recall', 'F1', 'MAP']
```

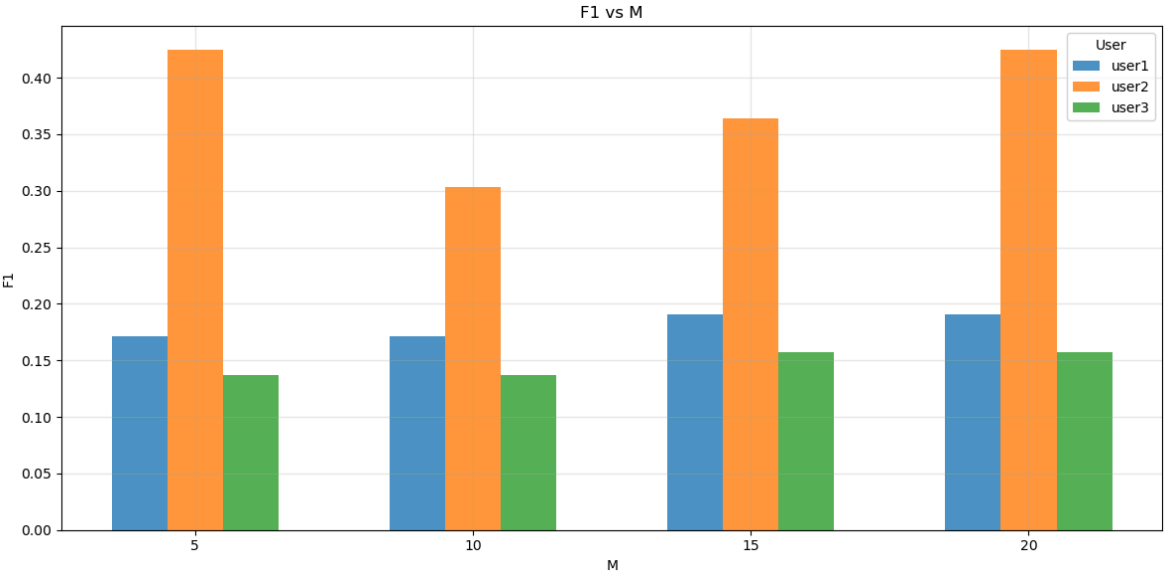
```
    M_list = sorted(df_numeric['M'].unique())
    user_list = df_numeric['User'].unique()

    bar_width = 0.2
    index = np.arange(len(M_list))

    for metric in metrics:
        plt.figure(figsize=(12, 6))

        for i, user in enumerate(user_list):
            user_data = df_numeric[df_numeric['User'] == user]
            user_data = user_data.sort_values('M')
            values = user_data[metric].values
            bar_positions = index + i * bar_width
            plt.bar(bar_positions, values, bar_width, label=user, alpha=0.8)

        plt.title(f'{metric} vs M')
        plt.xlabel('M')
        plt.ylabel(metric)
        plt.xticks(index + bar_width * (len(user_list) - 1) / 2, M_list)
        plt.legend(title='User')
        plt.grid(True, alpha=0.3)
        plt.tight_layout()
        plt.show()

        if not df_all.empty:
            plt.figure(figsize=(10, 6))
            for i, user in enumerate(user_list):
                user_all_data = df_all[df_all['User'] == user]
                if not user_all_data.empty:
                    value = user_all_data[metric].iloc[0]
                    plt.bar(i, value, label=user, alpha=0.8)

            plt.title(f'{metric} for M=all')
            plt.xlabel('User')
            plt.ylabel(metric)
            plt.xticks(range(len(user_list)), user_list)
            plt.legend(title='User')
            plt.grid(True, alpha=0.3)
            plt.tight_layout()
            plt.show()

plot_metrics_comparison(df_result)
```

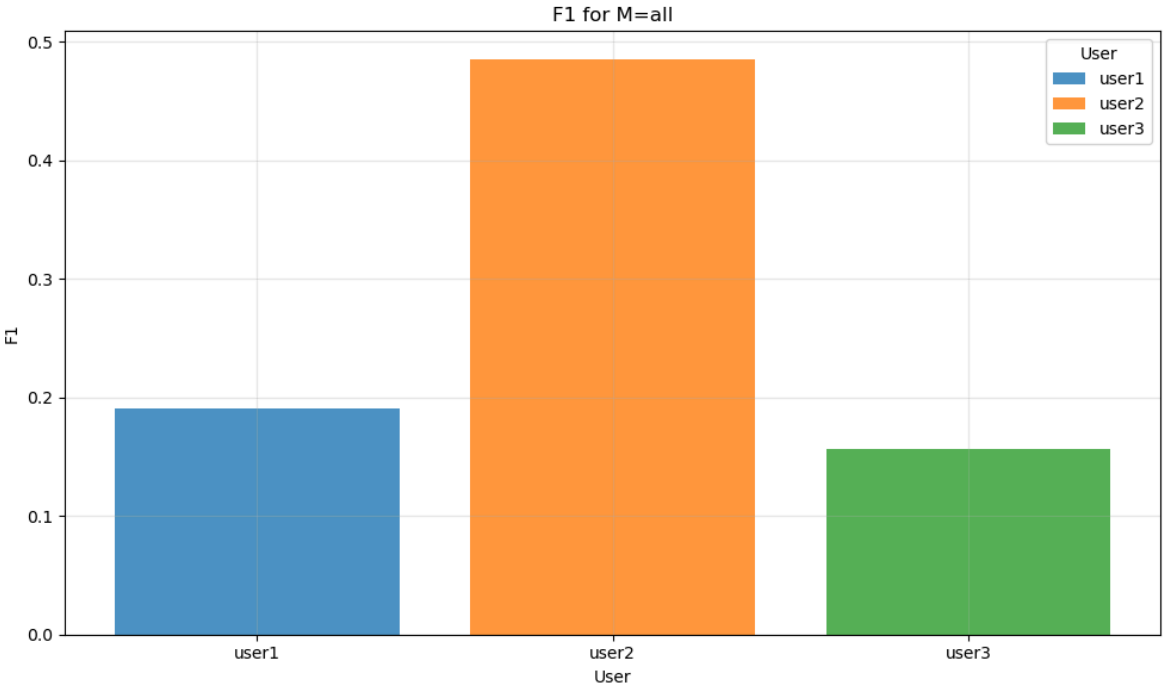

Precision vs M
User
user1
user2
user3
Precision
M
0.0
0.2
0.4
0.6
0.8
1.0
5
10
15
20

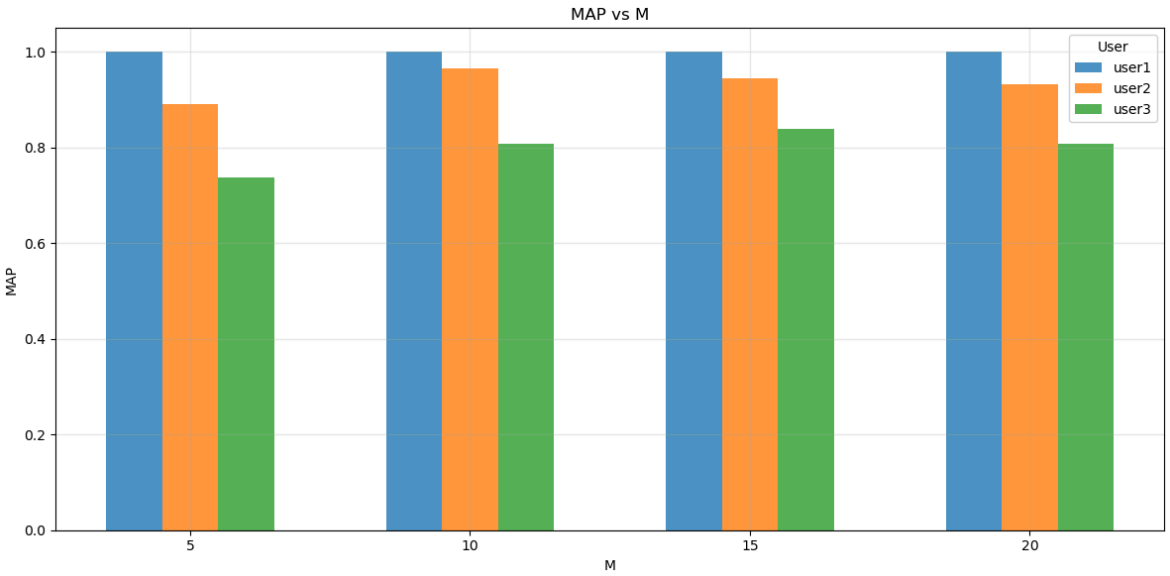

Precision for M=all
User
user1
user2
user3
Precision
User
0.0
0.2
0.4
0.6
0.8
1.0

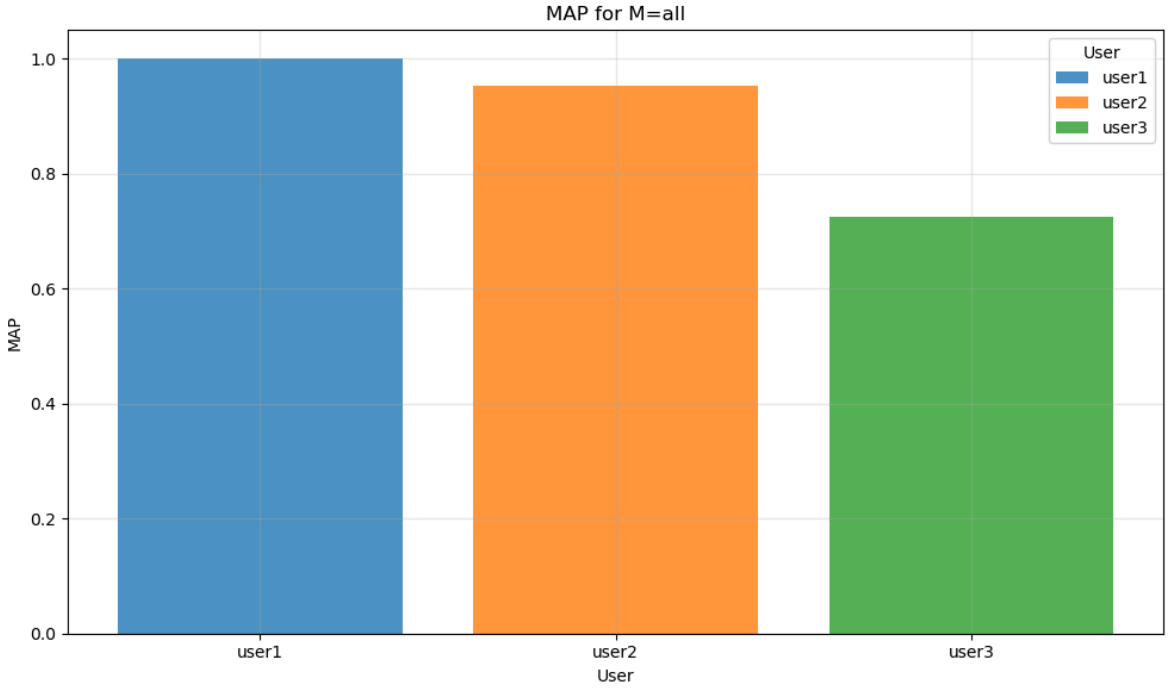

Recall vs M
User
user1
user2
user3
Recall
M
0.00
0.05
0.10
0.15
0.20
0.25
0.30
5
10
15
20


Recall for M=all
User
user1
user2
user3
Recall
0.35
0.30
0.25
0.20
0.15
0.10
0.05
0.00
user1
user2
user3
User


F1 vs M
User
user1
user2
user3
F1
0.40
0.35
0.30
0.25
0.20
0.15
0.10
0.05
0.00
5
10
15
20
M


F1 for M=all
User
user1
user2
user3
F1
0.5
0.4
0.3
0.2
0.1
0.0
user1
user2
user3
User





Optimal M Selection

From the plots:

- **Precision** continues to be high for `M=15` and `M=20`, especially for user1 and user3.
- **Recall** slightly increases with larger `M`, up to an optimal of `M=20` for user2.
- **F1-score** (harmonic mean of precision and recall) best balance at `M=20`.
- **MAP** is highest and most stable for `M=15` and `M=20`, which indicates stable overall ranking quality.

I select `M=20` as the best since it achieves:

- High **precision** for all;
- Highest **recall** and **F1** for user2 (with broader interests);
- Consistently high **MAP**, which means that not just correct but highly ranked recommended items are given.

It reflects that a greater number of top keywords can be covered (up to 20) which allows better coverage of users' interests without compromising recommendation quality.

Another matching algorithm: weighted keyword matching

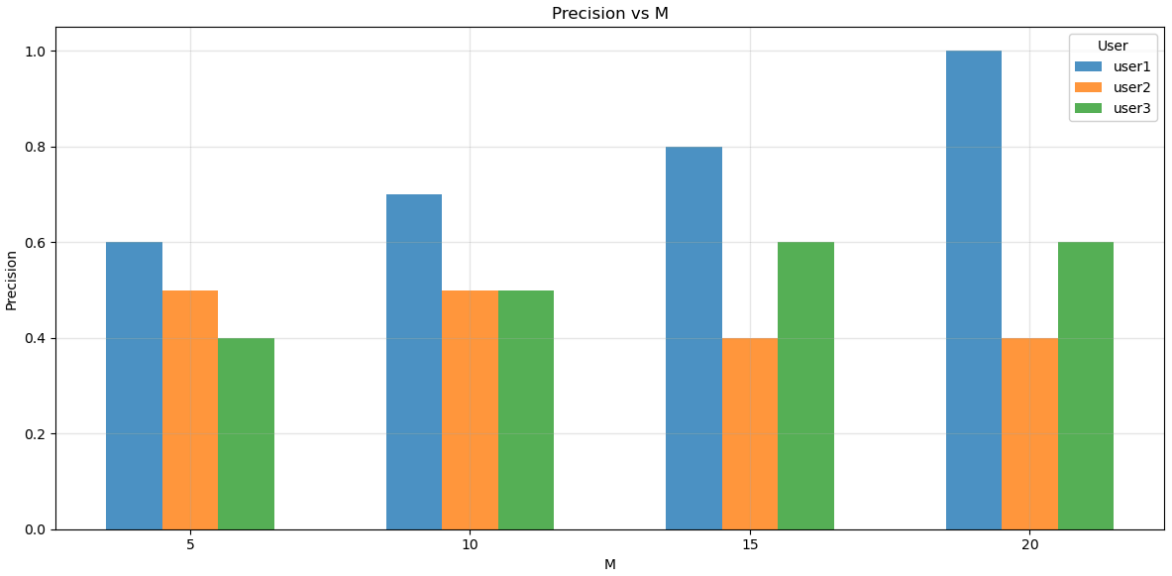
In [29]:
```
def build_user_profile(user_keywords_dict, df, vec, top_m='all', repeat_times=50

    from sklearn.preprocessing import normalize
    import numpy as np
    from scipy.sparse import csr_matrix

    profile = {}

    for topic, keywords in user_keywords_dict.items():

        topic_songs = df[df['topic'] == topic]['clean_lyrics'].tolist()

        if not topic_songs:
            continue

        topic_vec = vec.transform(topic_songs)

        avg_vec = topic_vec.mean(axis=0)

        feature_names = vec.get_feature_names_out()

        if top_m == 'all':
            selected_topic_words = feature_names
        else:
            arr = avg_vec.A[0] if hasattr(avg_vec, 'A') else avg_vec.toarray()[0
            top_idx = arr.argsort()[::-1][:int(top_m)]
            selected_topic_words = [feature_names[i] for i in top_idx]

        keywords_list = list(keywords) if isinstance(keywords, set) else keyword
        user_keywords = [kw for kw in keywords_list if kw in feature_names]

        if not user_keywords:
            continue

        enhanced_texts = []

        enhanced_texts.extend(topic_songs)

        for keyword in user_keywords:
            repeated_keyword = ' '.join([keyword] * repeat_times)
            enhanced_texts.append(repeated_keyword)

        enhanced_vec = vec.transform(enhanced_texts)

        avg_vec_enhanced = enhanced_vec.mean(axis=0)

        if top_m != 'all':
            arr_enhanced = avg_vec_enhanced.A[0] if hasattr(avg_vec_enhanced, 'A

            masked = np.zeros_like(arr_enhanced)
            for word in selected_topic_words:
                if word in feature_names:
                    idx = list(feature_names).index(word)
```

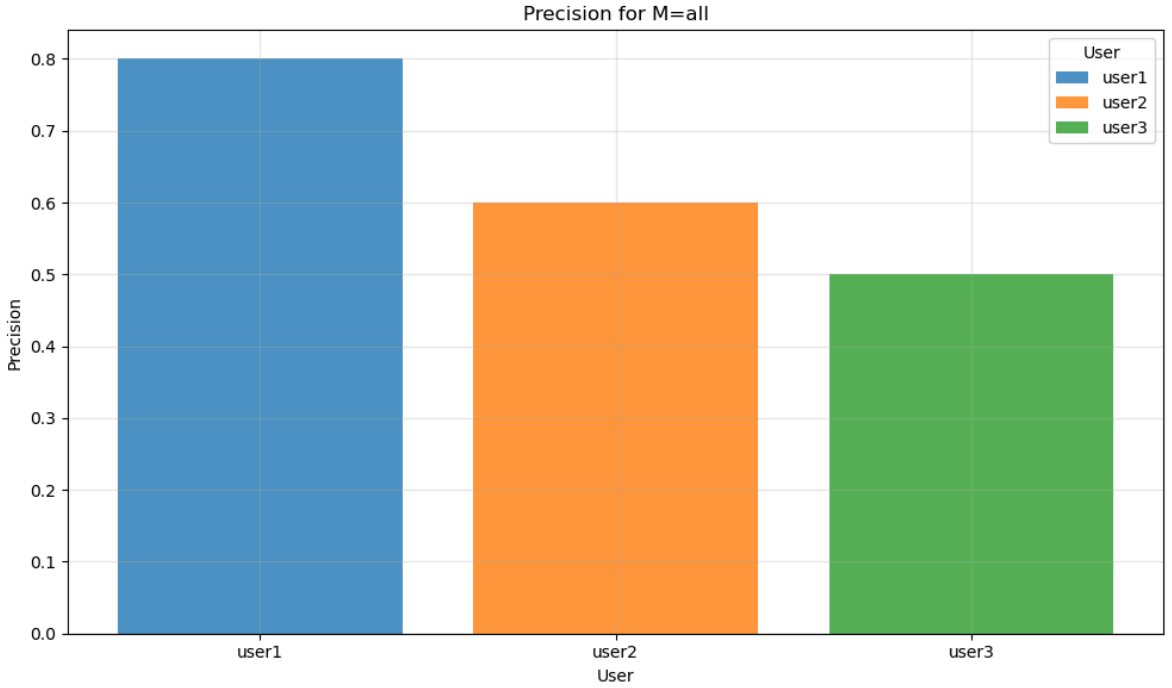
```
                    masked[idx] = arr_enhanced[idx]

            profile_vec = csr_matrix(masked.reshape(1, -1))
        else:
            profile_vec = avg_vec_enhanced

        if hasattr(profile_vec, 'toarray'):
            profile_vec_array = profile_vec.toarray()
        elif hasattr(profile_vec, 'A'):
            profile_vec_array = profile_vec.A
        else:
            profile_vec_array = np.asarray(profile_vec)

        if profile_vec_array.ndim == 1:
            profile_vec_array = profile_vec_array.reshape(1, -1)

        profile_vec_normalized = normalize(profile_vec_array, norm='l2')
        profile[topic] = profile_vec_normalized

    return profile
```

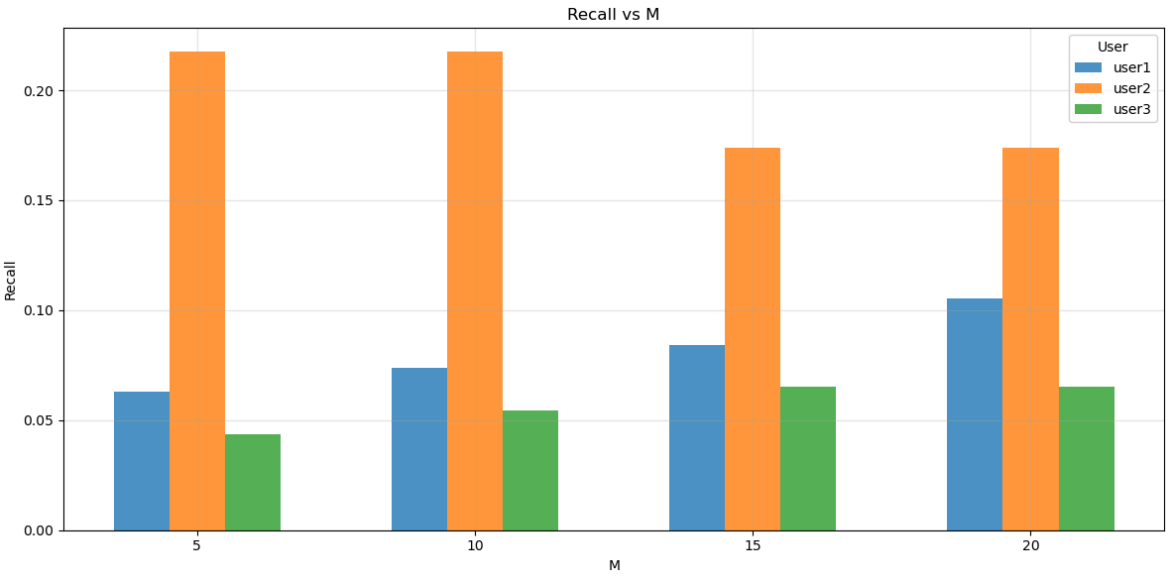
In [30]:

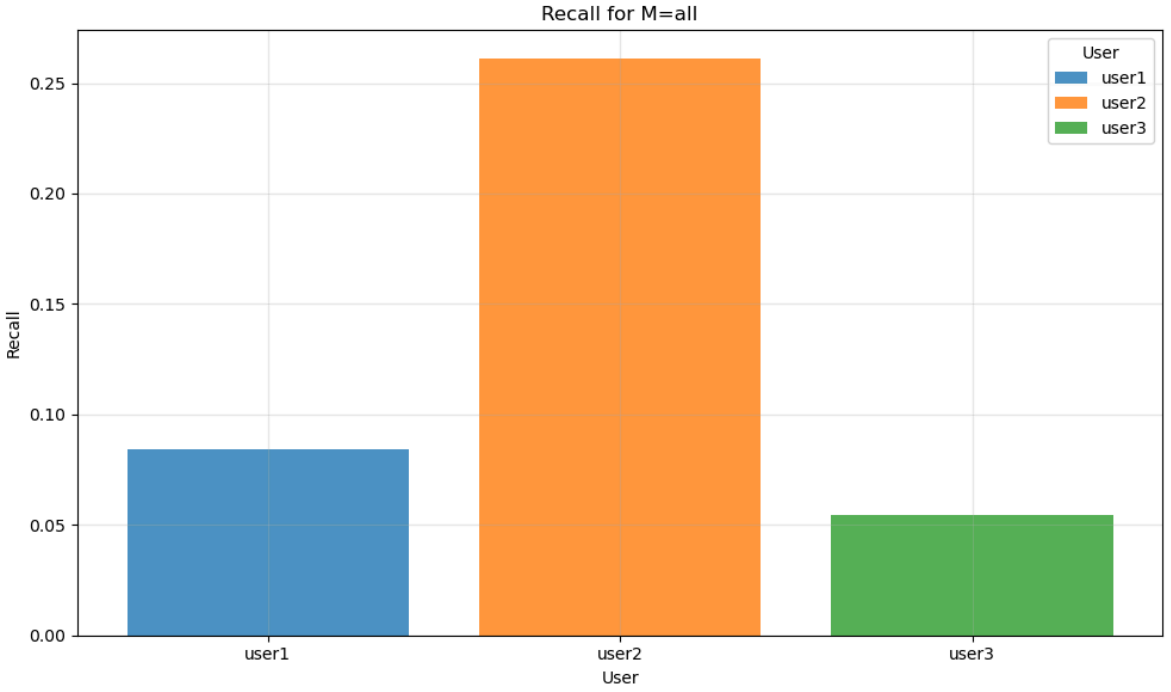
```
import warnings
from scipy.sparse import SparseEfficiencyWarning
warnings.filterwarnings('ignore', category=SparseEfficiencyWarning)

def final_evaluation(M_list, N, user_liked_dicts, train_data, test_data,
                                    best_vectorizer_tfidf, X_test_vec_tfidf):

    results = []

    for M in M_list:
        for uid, (liked_dict, liked_songs) in user_liked_dicts.items():

            user_profile = build_user_profile(liked_dict, train_data, best_vecto

            sim_scores = compute_similarity_scores(user_profile, X_test_vec_tfid

            topN_df = get_top_n_recommendations(test_data, sim_scores)

            precision, recall, f1, map = evaluate_recommendations(topN_df, liked

            results.append((uid, M, precision, recall, f1, map))

    df_result = pd.DataFrame(results, columns=['User', 'M', 'Precision', 'Recall

    return df_result


M_list = [5, 10, 15, 20, 'all']
N = 10
user_liked_dicts = {
    'user1': (user1_keywords, user1_liked),
    'user2': (user2_keywords, user2_liked),
    'user3': (user3_keywords, user3_liked)
}

df_result = final_evaluation(
    M_list=M_list,
    N=N,
    user_liked_dicts=user_liked_dicts,
```

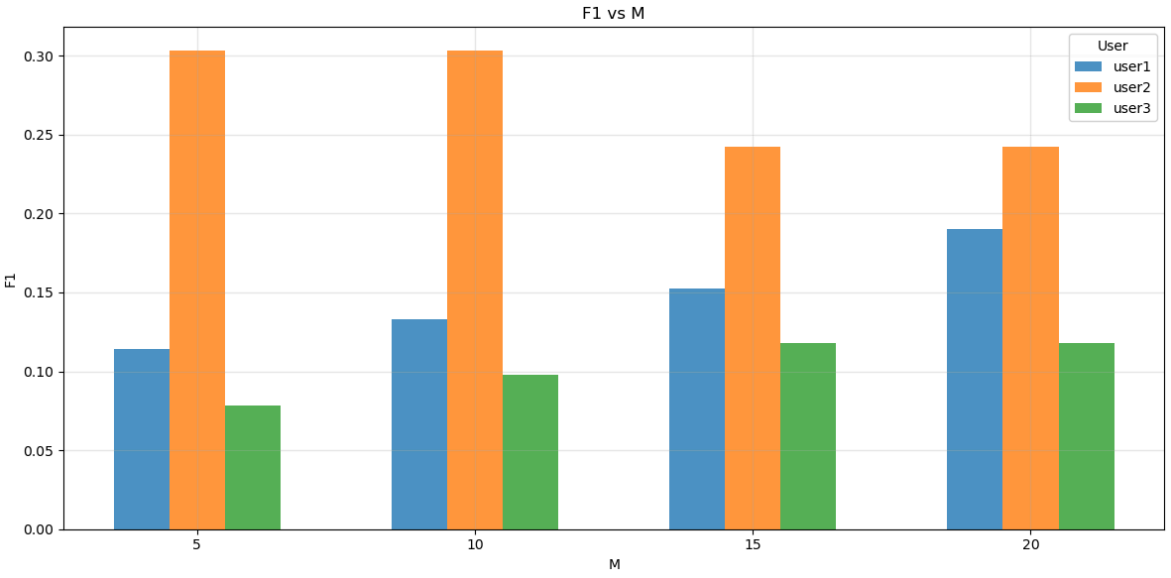
```
    train_data=train_data,
    test_data=test_data,
    best_vectorizer_tfidf=best_vectorizer_tfidf,
    X_test_vec_tfidf=X_test_vec_tfidf
)
plot_metrics_comparison(df_result)
```

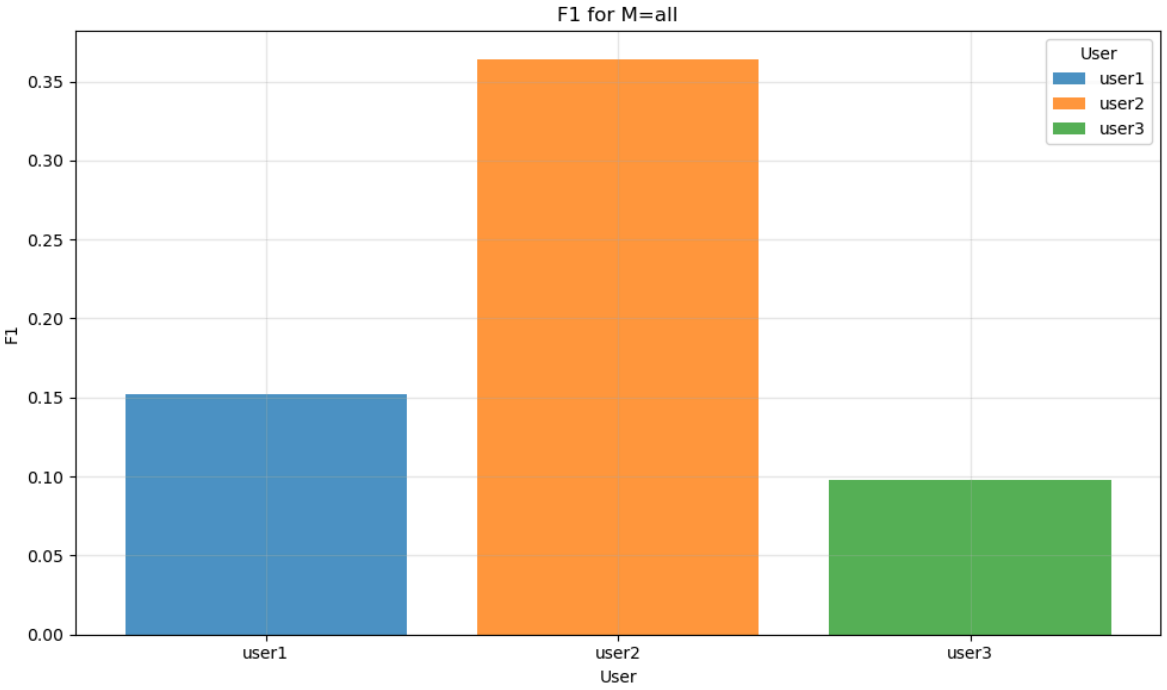


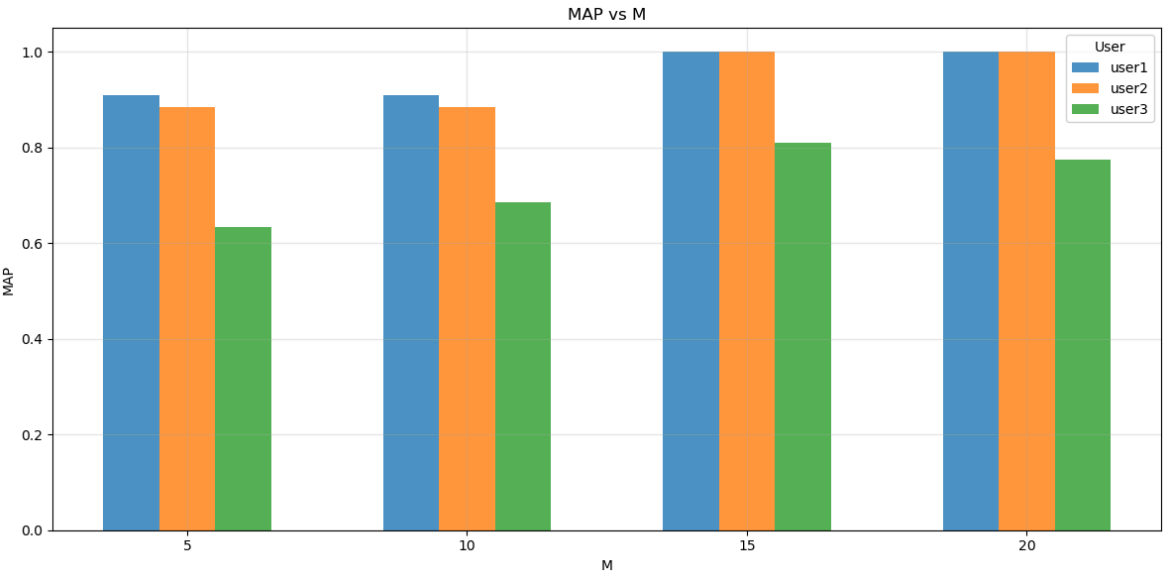


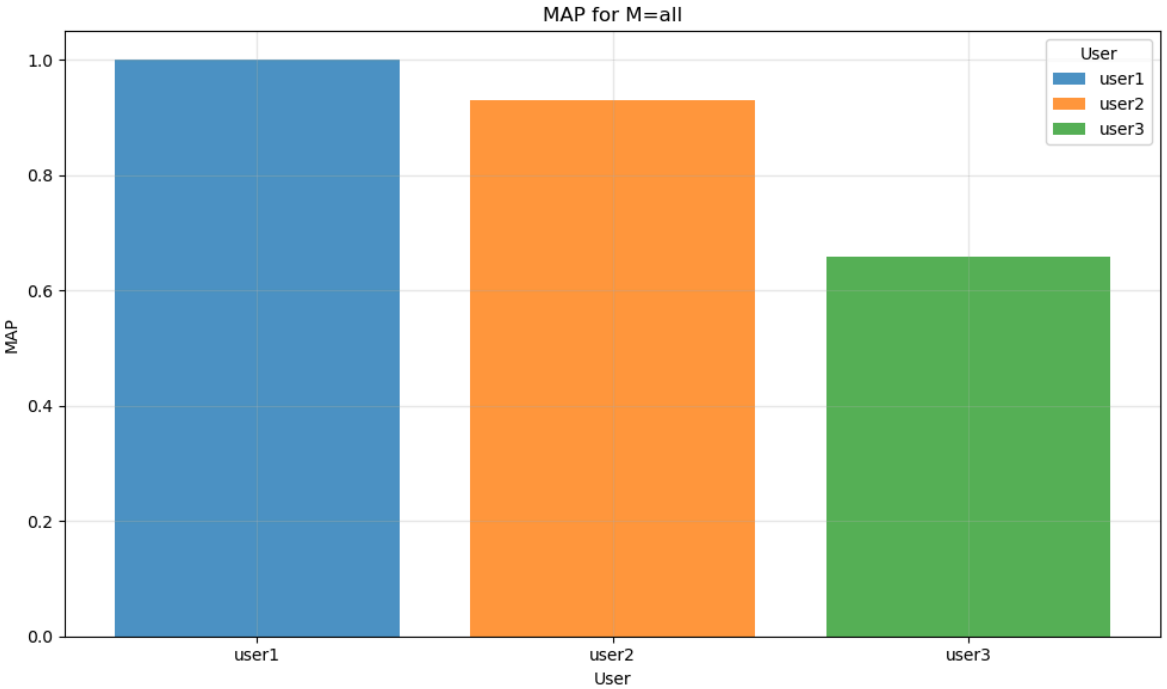

Recall vs M
User
user1
user2
user3
Recall
0.00
0.05
0.10
0.15
0.20
5
10
15
20
M


Recall for M=all
User
user1
user2
user3
Recall
0.00
0.05
0.10
0.15
0.20
0.25
user1
user2
user3
User


F1 vs M
User
user1
user2
user3
F1
0.00
0.05
0.10
0.15
0.20
0.25
0.30
5
10
15
20
M


F1 for M=all
User
user1
user2
user3
F1
0.00
0.05
0.10
0.15
0.20
0.25
0.30
0.35
user1
user2
user3
User


MAP vs M
User
user1
user2
user3
MAP
0.0
0.2
0.4
0.6
0.8
1.0
5
10
15
20
M


MAP for M=all
User
user1
user2
user3
MAP
0.0
0.2
0.4
0.6
0.8
1.0
user1
user2
user3
User

Matching Algorithm Comparison and Selection

For this project, I experimented with two content matching algorithms:

- **Full-text TF-IDF Cosine Similarity**: Creates user profiles based on the full-text lyrics of songs a user "likes", and computes cosine similarity between the user profile and candidate songs using TF-IDF vectors.
- **Weighted Keyword Matching**: Uses user-provided topic keywords to generate pseudo-documents. The keywords are copied to increase their TF-IDF weights, building a focused user profile that is then used for similarity matching.

Comparative Analysis:

- **Accuracy**: Both methods are similar in terms of Precision and MAP, but the weighted keyword method tends to overfit some themes.
- **Diversity**: Full-text TF-IDF considers more contextual information, which results in more diverse recommendation outputs.
- **Stability and Generalization**: The keyword-based approach is highly sensitive to the quality and quantity of input keywords, so it is less stable. By comparison, the full-text approach generalizes better to different structures of user interests.
- **Interpretability**: The keyword-based approach is more interpretable because user profiles directly represent interest keywords.

Therefore, the **most comprehensive and strongest is the full-text TF-IDF cosine similarity method**. It was ultimately chosen as the primary recommendation algorithm due to its ability to model user interests nicely and at the same time offer broader coverage and scalability.

Part3

Week1-3: Randomly recommended sample pool

In [37]:
```
import pandas as pd
import random

df = pd.read_csv("dataset.tsv", sep="\t")

week1_df = df.iloc[0:250].copy()
week2_df = df.iloc[250:500].copy()
week3_df = df.iloc[500:750].copy()
week4_df = df.iloc[750:1000].copy()

N = 10
# Remove random_state to ensure different results each run
week1_random = week1_df.sample(N)
week2_random = week2_df.sample(N)
week3_random = week3_df.sample(N)
```

In [38]:

```
user_selections = []

# Week 1 recommendations
print("=== Week 1 Recommended Songs ===")
for idx, row in week1_random.iterrows():
    print(f"Index {idx}: {row['artist_name']} - {row['track_name']} - {row['rele

print("\nEnter the index of songs you like (type 'done' to finish):")
while True:
    user_input = input().strip()
    if user_input.lower() == 'done':
        break
    try:
        song_index = int(user_input)
        if song_index in week1_random.index:
            user_selections.append(song_index)
            print(f"Selected: Index {song_index}")
        else:
            print("Invalid index, please try again")
    except ValueError:
        print("Please enter a valid number or 'done'")

# Week 2 recommendations
print("\n=== Week 2 Recommended Songs ===")
for idx, row in week2_random.iterrows():
    print(f"Index {idx}: {row['artist_name']} - {row['track_name']} - {row['rele

print("\nEnter the index of songs you like (type 'done' to finish):")
while True:
    user_input = input().strip()
    if user_input.lower() == 'done':
        break
    try:
        song_index = int(user_input)
        if song_index in week2_random.index:
            user_selections.append(song_index)
            print(f"Selected: Index {song_index}")
        else:
            print("Invalid index, please try again")
    except ValueError:
        print("Please enter a valid number or 'done'")

# Week 3 recommendations
print("\n=== Week 3 Recommended Songs ===")
for idx, row in week3_random.iterrows():
    print(f"Index {idx}: {row['artist_name']} - {row['track_name']} - {row['rele

print("\nEnter the index of songs you like (type 'done' to finish):")
while True:
    user_input = input().strip()
    if user_input.lower() == 'done':
        break
    try:
        song_index = int(user_input)
        if song_index in week3_random.index:
            user_selections.append(song_index)
            print(f"Selected: Index {song_index}")
        else:
            print("Invalid index, please try again")
```

```
    except ValueError:
        print("Please enter a valid number or 'done'")

user_selections = list(set(user_selections))

# Build training dataset
print(f"\nTotal selected: {len(user_selections)} songs")
training_data = df.loc[user_selections].copy()
print("\n=== Training Dataset Song Information ===")
for idx, row in training_data.iterrows():
    print(f"Index {idx}: {row['artist_name']} - {row['track_name']} - {row['rele

print(f"Training dataset size: {len(training_data)} songs")
```

=== Week 1 Recommended Songs ===
Index 228: sleeping wolf - new kings - 2017 - rock - stand fight need mean second guessin yeah gonna wear crown cause stand fight stop kings ignore like pass raise voice scar scar power come ignore stand fight need lyric commercial - dark
Index 112: madeleine peyroux - shout sister shout - 2016 - jazz - shout sister shout hallelujah shout 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 sister shout tell world brilliant fool heaven observe golden rule sweetheart wife quit baby lose life shout sister shout hallelujah shout sister shout hallelujah shout sister shout yeah tell world reason mountain reason reason doctor give patient pill reason dance reason sing reason band swing understand bird understand be understand 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 - dark
Index 230: dirty heads - visions (featuring chloe chaidez of kitten) - 2018 - reggae - mind know want plan offer try help load little lighter share wealth good luck good luck know problems eye want live morning brother surprise good luck good luck yeah yeah yeah yeah yeah yeah yeah yeah yeah time go receive time want believe breathe good luck good luck good luck good luck - emotion
Index 138: ballyhoo! - drink about you - 2017 - reggae - drink life layin spend nights stalk picture crew someday soon right good thing heart sing fool cause tool eye burn like cause liar say love destroy days know stop fallin drink life layin spend nights stalk picture crew someday soon right derail train insane candycoated thoughts brain loser drown booze cause stand pain days know stop fallin drink life layin spend nights stalk picture crew someday soon right gotta like high high girl dream waste days know stop fallin drink life life layin spend nights spend nights stalk picture crew someday soon right right drink life life layin spend nights stalk picture crew crew someday soon right - personal
Index 24: t-rock - be a g about it - 2016 - hip hop - 잊어버리지마 잃어버리지마 잊어버리지마 잃어버리지마 멈췄으면 잊어버리지마 잃어버리지마 잊어버리지마 잃어버리지마 바라보는 영원하길 기억해주길 한번쯤은 돌아보길 바라봐주길 baby 잊어버리지마 잊어버리지마 잊어버리지마 romanization neowa eonjenga nami dwieodo yeongyeong daheul eomneun doeeodo ijeobeorijima ilheobeorijima hoksina dareun saramui japgo isseodo yeongyeong daheul eomneun gose isseodo ijeobeorijima ilheobeorijima ttatteuthae bori matdaheul ttae salmyeosi sirin jabajul ttae chagaun sesange jichin mameul ongiro gamssane nune damgin neoui eolgul pume jamdeun neoui moseup idaero kkwak jabeun nohchimyeon andwae neowa eonjenga nami dwieodo yeongyeong daheul eomneun doeeodo ijeobeorijima ilheobeorijima hoksina dareun saramui japgo isseodo yeongyeong daheul eomneun gose isseodo ijeobeorijima ilheobeorijima maju neol baraboneun jigeumi yeongwonhagil barae hoksina sigani uril jiltu halkka sasil geokjeong dwae byeonchi yeongwonhi hamkke eonjenga gieokhaejugil hanbeonjjeumeun dorabogil budi nochi marajwo uril gyesok barabwajugil sigani jina baby neowa uriga nami dwieodo ijeobeorijima ijeobeorijima norael ijeobeorijima english estrange someday meet forget lose hold hand forget lose warm cheek meet hold freeze hand gently hug heart wear cold world warmth face eye sleep arm stop like hand hold estrange someday meet forget lose hold hand forget lose moment sit facetoface look forever time envy actually worry change forever remember someday look look forever time go baby estrange someday forget forget forget song - personal
Index 121: yungblud - hope for the underrated youth - 2019 - rock - leave today throw away magazine hair change second juvenile stay heart better sorry better late heart better hide feel know dream cause underrate youth tell truth cause pull pull pull yeah underrate youth episode like fiction mix true lyric commercial - sadness
Index 197: eric ethridge - if you met me first - 2018 - country - strangest creature fail prey like vulture hand follow follow follow life steal light sweet taste talk wanna face crowd friend failure trust gods couldn felt lose steal light sweet taste talk wanna face crowd steal light sweet taste talk wanna face crowd - dark

Index 115: eric clapton - layla (with j. j. cale) - 2016 - blues - help surrender control search death soul bleed beneath cover keep leak bleed inside fear fall fe et time lose silence reach life star save need guilty feel come crash take help l ose lock cage hate take control break bleed inside fear fall feet time lose silen ce reach life star save need guilty feel come crash take silence reach life star save need jury bury come crash take take silence reach life star save need guilty feel come crash take silence take save need guilty feel come crash take - dark
Index 137: luke combs - beer never broke my heart - 2019 - country - largemouth b ass bust line couple beautiful girls tell goodbye truck break dog politicians fir e boss take hand count things count longneck icecold beer break heart like ring f ootball team tear apart like neon dream dawn bar guitar longneck icecold beer bre ak heart blue baby eye drive crazy taillights leave know sure know longneck iceco ld beer break heart like ring football team tear apart like neon dream dawn bar g uitar longneck icecold beer break heart take hand count things count hand grippin cold cause longneck icecold beer break heart like ring football team tear apart l ike neon dream dawn bar guitar longneck icecold beer break heart break heart - sa dness
Index 7: thank you scientist - the amateur arsonist's handbook - 2016 - jazz - qu ick think good true worst best things forever pessimist drag complicate feel fear tell reason nice slow tell tell reason reason yeah hell understand lose win hand burn walk away hand satisfy notion break justify evidence lose things pretend par don earth search solid grind fear tell reason nice slow tell tell reason reason y eah hell understand lose win hand burn walk away hand hear word black white mean hear hear word black white mean hear say ohhh say leave gonna leave poison say sa y gonna leave poison say leave gonna leave poison ohhh ohhh ohhh hell hell unders tand lose win hand burn walk away hand ohhhhhhhhhhh ohhhhhhhhhh - dark

Enter the index of songs you like (type 'done' to finish):
Selected: Index 228
Selected: Index 112
Selected: Index 230
Selected: Index 121

=== Week 2 Recommended Songs ===
Index 464: kelsea ballerini - graveyard - 2018 - country - wanna skeleton closet throw cause know okay stone road forget cause fall fast dash date wanna heart gra veyard cold hard dirt throw wanna watch drive away black hopeless break girl litt le black dress break heart rest peace right maybe naive believe hurt know hide da rkest suit shade know hand shoe dirty guess need shovel grave wanna heart graveya rd cold hard dirt throw wanna watch drive away black hopeless break girl little b lack dress break heart rest peace right yeah gonna night cause ghost break heart stay feet yeah gonna night cause ghost break heart stay feet wanna heart graveyar d cold hard dirt throw wanna watch drive away black hopeless break girl little bl ack dress break heart rest peace right hopeless break girl break heart rest peace right wanna heart wanna heart graveyard babe graveyard - sadness
Index 363: mandisa - bleed the same - 2017 - rock - bleed beautiful come bleed te ll tell divide wake today headline innocent life take hatred hard think good mist ake cause heart break tell leave right point finger take side gonna realize bleed beautiful come bleed tell tell divide gonna fight fight gonna shout bleed tell te ll divide tell judge kind clothe wear color skin black black white white aren ins ide inside father open eye bleed bleed beautiful come bleed bleed tell tell divid e gonna fight fight fight gonna shout bleed bleed tell tell divide drive darkness fight say drive darkness fight bleed bleed beautiful come stand unite bleed bleed tell tell divide gonna fight fight fight gonna shout bleed bleed bleed stand unit e stand unite stand unite father pray families come right seek face forgive sin h eal incredible land savior amen - dark
Index 303: banners - start a riot - 2016 - rock - march street like ship surrende r retreat tear wall warm bring home burn city dust safe sound leave trust hand st and world fall apart start riot night fall heart light dark sound alarm arm start riot smoke like sunlight haze fight till flag wave white die days bomb blast worl

d fall apart start riot night fall heart light dark sound alarm arm start riot st art start riot start start riot start start riot start start riot world fall apar t start riot - sadness
Index 426: seedless - it's always something - 2016 - reggae - know hard mind know tough leave ones people change hear start believe feel sicker hit everyday allow souls wither turn blue sky liquor smoke days away stray sleepless nights drunken fight couldn toss aside selfdestructive pride grow life tough fall luck know hard mind know tough leave ones toxic know go live like toxic wonder amiss sleepless n ights drunken fight couldn toss aside selfdestructive pride grow life tough fall luck separate paths go life guess separate paths look feel people change hear sta rt believe people change grow hear work start believe people change burn jade mis place allegations hear think time start believe move people change grow hear work start believe people change burn jade misplace allegations hear think time start believe move people change grow hear work start believe people change burn jade m isplace allegations hear think time start believe move - personal
Index 373: ky-mani marley - rule my heart - 2016 - reggae - moment time like know kymani hide life believe beauty stand right eye girl rule heart girl rule heart g irl rule heart girl rule heart rare special kind world mind sight baby hold close night girl rule heart girl rule heart girl rule heart girl rule heart girl fantas ize body hypnotize run circle mind yeah wanna treat right wait life girl divine g irl rule heart girl rule heart girl rule heart girl rule heart girl rule heart gi rl rule heart girl rule heart girl rule heart - sadness
Index 411: filmore - heart's having a hard time - 2018 - country - think weight c hest night say couldn think break wouldn better start think easy hat cause afraid hurt think good goodbye heart have hard time hard time make sense right inside ye ah know mind heart heart have hard time shouldn tough change lock screen sink kis s smile million reason leave think feel feel hat cause afraid hurt think good goo dbye heart have hard time make sense right inside yeah know mind heart heart have hard time miss wish work wrong love hat cause afraid hurt think good goodbye hear t have hard time make sense right inside yeah know mind heart heart have hard tim e heart have hard time - sadness
Index 372: greensky bluegrass - past my prime - 2016 - rock - twentyseven dollars jacket dust collar walk hours say like long television heart ambition haunt dream reach grander things know past prime look reason time bottle list chance miss vac ant place promise escape drink useless mess helpless self defense look past prime look reason time knees aren bleed defeat save face swear - personal
Index 285: welshly arms - sanctuary - 2018 - blues - darkness sleep hold close sp ace need spark go glow bring home yeah bring home hurt feel pain dirt wash rain w alk road felt shame place home time changin sanctuary shelter peace sanctuary saf e sanctuary shelter peace sanctuary safe rain start toll slow cause know world tu rn cold need hold hold hold hurt feel pain dirt wash rain walk road felt shame pl ace home time changin sanctuary shelter peace sanctuary safe sanctuary shelter pe ace sanctuary safe yeah yeah safe yeah yeah share hurt share pain dirt wash rain walk road felt shame time changin sanctuary hold sanctuary hold safe sanctuary sh elter peace sanctuary safe sanctuary shelter peace sanctuary safe yeah yeah safe yeah yeah safe sanctuary safe hold cause sanctuary - sadness
Index 305: brooks & dunn - red dirt road (with cody johnson) - 2019 - country - d ecisions desperation internal instincts crave isolation grow fear come alive plac e die demons dream know need realign fell river illusion apathy drown selfinduced confusion yeah fear come alive place die demons dream know need realign fear come alive place die demons dream know need realign fear come alive place die demons d ream know need realign fear come alive place die demons dream know need realign f ear come alive place die demons dream know need realign yeah - dark
Index 258: pepper - the invite - 2016 - reggae - drive check engine light right p lan think drive squander time lock inside mind talk dream afraid fail routine act ion happen wait talk knock kick door light alright whoa yeah welcome life live wa nt live whoa yeah welcome life cause right life alive need begin stick indecision people lose look like risk reward tell jump train think think action happen wait stop want kick door roll dice alright whoa yeah welcome life live want live whoa yeah welcome life cause life life alive alive whoa yeah welcome life live want li

ve whoa yeah welcome life cause life life whoa yeah welcome life live want live w
hoa yeah welcome life cause life cause life life alive - personal

Enter the index of songs you like (type 'done' to finish):
Selected: Index 426
Selected: Index 373
Selected: Index 372
Selected: Index 285
Selected: Index 258

=== Week 3 Recommended Songs ===
Index 554: all them witches - rob's dream - 2018 - blues - dagger sword dagger sw
ord dark groves call dark groves call dagger sword dagger sword dark groves call
dark groves call turn blue turn blue turn blue turn blue turn blue turn blue need
le heart stop needle heart stop sinkin grind feet riptide sinkin grind food ripti
de turn blue turn blue turn blue turn blue turn blue turn blue turn blue - lifest
yle
Index 620: russ - psycho, pt. 2 - 2016 - pop - go psycho go live tightrope go go
psycho go live tightrope go know like want tonight pick band feelin yeah go psych
o go live tightrope go go psycho go live tightrope go know know special hook in
strumental reiterate commitment explore like ride like horse like interest baby i
nfatuate hold graduation help cash steer clear evasion death grip assassination g
et carry away marry today like whitesnake crazy crazy break open possibility h
oe cash want open possibility hoe cash go psycho go live tightrope go go psycho g
o live tightrope go go psycho - emotion
Index 567: greta van fleet - brave new world - 2018 - blues - drifters high rift
plain ash acid rain turn dust eye choke death smog lie look sky darkness realize
kill fear power lie hypnotize turn clock glass sand time blacken land silent chil
d climb mound char plant seed grow star look sky darkness realize kill fear power
lie hypnotize look sky darkness realize kill fear power lie hypnotize look sky da
rkness realize kill fear power lie hypnotize - dark
Index 628: ed sheeran - what do i know? - 2017 - pop - soapbox stand give stage g
uitar song daddy tell involve politics religions people quarrel paint picture sce
ne know children know mean pass things family give understand positivity change w
orld piano bass guitar grab beat away oneman university degree lord know everybod
y talk bout exponential growth stock market crash portfolios sit song write sing
change world moment know change world moment know change world moment revolution
come minute away people march streets today know hate balance razor blade paint p
icture scene know people follow dream reremember life fittin jeans understand pos
itivity change world piano bass guitar grab beat away oneman university degree lo
rd know everybody talk bout exponential growth stock market crash portfolios sit
song write sing change world moment know change world moment know change world mo
ment paint picture scene know future hand free spread understand positivity chang
e world piano bass guitar grab beat away oneman university degree lord know every
body talk bout exponential growth stock market crash portfolios sit song write si
ng change world moment know change world moment know change world moment - person
al
Index 614: ivan ave - squint - 2017 - jazz - awake stock fridge heart pad wall pa
per brain wall suede beanbag sit mind spiritual interior design squint swear dem
ons door look like fear squadded anymore long squint mirror eagle flow room plu
ck seagull throw bricks build cathedral time build people mutual vibes rely brew
doobies stop live future time tryna lie squint stare hard noise corner dark cloud
clean house stare hard squinty eye liftin blind look like lately days tape self
forget rewind come long eye tunnel long analyse stairs kiss forehead mother turn
hors oeuvres worm ash coldlamping fireplace grind away high stake shout hear cur
ve lobster butter healer mutual bout feast like squeeze cheek niece squint stare
hard noise corner dark cloud stare hard stare dark - dark
Index 599: zayn - good years - 2018 - pop - close eye crowd thousand tear pray wa
ste good years good years good years voice scream loud hell care bout world bring
high star worry want sorry close eye crowd thousand tear pray waste good years go

od years good years drug alcohol hell fight cause world know numb dumb change st
ory want sorry close eye crowd thousand tear pray waste good years good years goo
d years need breathe feel alive meet night light feel wind hold pain deep inside
eye eye close eye crowd thousand tear pray waste good years good years good years
pray waste good years good years good years - personal
Index 626: nahko and medicine for the people - runner - 2016 - reggae - runner ru
n things good things ahead kind lover follow dragonflies lead sunrise morning tid
e even yeah teach harder lessons yeah kiss softly kiss softly place know best who
aohohoh whoaohohohohoh whoaohohoh yeah whoaohohoh whoaohohohohoh whoaohohoh yeah
loyal return family earn string leash clip wing stronger things nature want promi
se log hours maybe open mind open kiss softly kiss softly place know best return
return return place learn return return return place learn whoaohohoh whoaohohoho
hoh whoaohohoh yeah whoaohohoh whoaohohohohoh whoaohohoh yeah kiss softly kiss so
ftly place know best yeah kiss softly kiss softly place know best yeah waipio val
ley peakin tetons follow klamath basin soul shinin shasta holy headwaters dear co
lumbia gorgeous sparkle eye creators callin number outlaw rest reason yeah cause
things know return return - personal
Index 589: the dillinger escape plan - limerent death - 2016 - jazz - amaze reinf
orce guard cause deep inside mistake forever haunt persecute thoroughly give fit
give fit world believe know hold breath count star vacant glass movement cease th
ink time freeze instead rest state think forever know hear fall aokay feel fine
smile feel alright feel fine shin feel fine feel fine smile amaze reinforce guard
cause deep inside mistake forever haunt persecute thoroughly give fit give fit g
ive fit give fit give fit give fit give fit give fit give fit give want give want give wan
t give want give want give want give want give want give want give want give want
give world believe world know - personal
Index 565: twenty one pilots - chlorine - 2018 - pop - little sippin straight chl
orine vibe slide beat chemical beat chemical leave save seat complete moment medi
cal moment medical sippin straight chlorine lovin tastin woah venom tongue depend
ent time poisonous vibrations woah help body runnin life runnin life sippin strai
ght chlorine vibe slide beat chemical beat chemical leave save seat complete mome
nt medical moment medical sippin straight chlorine fall formation woah plan escap
e wall confine rebel carnation woaaoh grow decay runnin life runnin life yeah run
nin life runnin life hide coat pocket keep rebel felt invincible wrap head differ
ent live lead body live lead line read incorrect say lead terrible flavor double
paper maker despise hate fight life like sippin straight chlorine vibe slide beat
chemical beat chemical leave save seat complete moment medical moment medical sip
pin straight chlorine vibe vibe vibe vibe beat chemical yeah vibe vibe vibe vibe
moment medical yeah sippin straight chlorine vibe vibe vibe vibe beat chemical ye
ah vibe vibe vibe vibe moment medical yeah sorry forget catch speed test like end
weather flag build house piece chemical build house piece chemical build house pi
ece chemical build house piece chemical - personal
Index 505: foster the people - static space lover - 2017 - rock - circuit liken g
listen start shift start shift morning know come ahhahhahhahh ahhahhahhahh long l
ong long long hang forever hold hold static space lovers fine circle place bet pr
aise lord static lips leave want float cause live constellation reach cause hold
forever ahhahhahhahh ahhahhahhahh goodbye long long long long hang forever hold h
old static space lovers fine circle long long hang forever hold hold static space
lovers fine circle - lifestyle

Enter the index of songs you like (type 'done' to finish):
Please enter a valid number or 'done'
Selected: Index 628
Selected: Index 599
Selected: Index 626
Selected: Index 505

Total selected: 13 songs

=== Training Dataset Song Information ===

Index 258: pepper - the invite - 2016 - reggae - drive check engine light right p lan think drive squander time lock inside mind talk dream afraid fail routine act ion happen wait talk knock kick door light alright whoa yeah welcome life live wa nt live whoa yeah welcome life cause right life alive need begin stick indecision people lose look like risk reward tell jump train think think action happen wait stop want kick door roll dice alright whoa yeah welcome life live want live whoa yeah welcome life cause life life alive alive whoa yeah welcome life live want li ve whoa yeah welcome life cause life life whoa yeah welcome life live want live w hoa yeah welcome life cause life cause life life alive - personal
Index 228: sleeping wolf - new kings - 2017 - rock - stand fight need mean second guessin yeah gonna wear crown cause stand fight stop kings ignore like pass raise voice scar scar power come ignore stand fight need lyric commercial - dark
Index 230: dirty heads - visions (featuring chloe chaidez of kitten) - 2018 - reg gae - mind know want plan offer try help load little lighter share wealth good lu ck good luck know problems eye want live morning brother surprise good luck good luck yeah yeah yeah yeah yeah yeah yeah yeah yeah time go receive time want belie ve breathe good luck good luck good luck good luck - emotion
Index 426: seedless - it's always something - 2016 - reggae - know hard mind know tough leave ones people change hear start believe feel sicker hit everyday allow souls wither turn blue sky liquor smoke days away stray sleepless nights drunken fight couldn toss aside selfdestructive pride grow life tough fall luck know hard mind know tough leave ones toxic know go live like toxic wonder amiss sleepless n ights drunken fight couldn toss aside selfdestructive pride grow life tough fall luck separate paths go life guess separate paths look feel people change hear sta rt believe people change grow hear work start believe people change burn jade mis place allegations hear think time start believe move people change grow hear work start believe people change burn jade misplace allegations hear think time start believe move people change grow hear work start believe people change burn jade m isplace allegations hear think time start believe move - personal
Index 505: foster the people - static space lover - 2017 - rock - circuit liken g listen start shift start shift morning know come ahhahhahhahh ahhahhahhahh long l ong long long hang forever hold hold static space lovers fine circle place bet pr aise lord static lips leave want float cause live constellation reach cause hold forever ahhahhahhahh ahhahhahhahh goodbye long long long long hang forever hold h old static space lovers fine circle long long hang forever hold hold static space lovers fine circle - lifestyle
Index 112: madeleine peyroux - shout sister shout - 2016 - jazz - shout sister sh out hallelujah shout sister shout shout sister shout tell world reason live reaso n die darn good reason woman start cry reason mole reason dimple reason simple sh out sister shout hallelujah shout sister shout hallelujah shout sister shout mmhm m yeah shout sister shout tell world brilliant fool heaven observe golden rule sw eetheart wife quit baby lose life shout sister shout hallelujah shout sister shou t hallelujah shout sister shout yeah tell world reason mountain reason reason doc tor give patient pill reason dance reason sing reason band swing understand bird understand be understand eat cheese understand understand cat rhythm understand s hout sister shout hallelujah shout sister shout hallelujah shout sister shout mmh mm hallelujah tell world know gonna shout shout sister go shout shout sister go s hout tell world - dark
Index 626: nahko and medicine for the people - runner - 2016 - reggae - runner ru n things good things ahead kind lover follow dragonflies lead sunrise morning tid e even yeah teach harder lessons yeah kiss softly kiss softly place know best who aohohoh whoaohohohohoh whoaohohoh yeah whoaohohoh whoaohohohohoh whoaohohoh yeah loyal return family earn string leash clip wing stronger things nature want promi se log hours maybe open mind open kiss softly kiss softly place know best return return return place learn return return return place learn whoaohohoh whoaohohoho hoh whoaohohoh yeah whoaohohoh whoaohohohohoh whoaohohoh yeah kiss softly kiss so ftly place know best yeah kiss softly kiss softly place know best yeah waipio val ley peakin tetons follow klamath basin soul shinin shasta holy headwaters dear co lumbia gorgeous sparkle eye creators callin number outlaw rest reason yeah cause things know return return - personal

```
Index 372: greensky bluegrass - past my prime - 2016 - rock - twentyseven dollars jacket dust collar walk hours say like long television heart ambition haunt dream reach grander things know past prime look reason time bottle list chance miss vac ant place promise escape drink useless mess helpless self defense look past prime look reason time knees aren bleed defeat save face swear - personal
Index 373: ky-mani marley - rule my heart - 2016 - reggae - moment time like know kymani hide life believe beauty stand right eye girl rule heart girl rule heart g irl rule heart girl rule heart rare special kind world mind sight baby hold close night girl rule heart girl rule heart girl rule heart girl rule heart girl fantas ize body hypnotize run circle mind yeah wanna treat right wait life girl divine g irl rule heart girl rule heart girl rule heart girl rule heart girl rule heart gi rl rule heart girl rule heart girl rule heart - sadness
Index 628: ed sheeran - what do i know? - 2017 - pop - soapbox stand give stage g uitar song daddy tell involve politics religions people quarrel paint picture sce ne know children know mean pass things family give understand positivity change w orld piano bass guitar grab beat away oneman university degree lord know everybod y talk bout exponential growth stock market crash portfolios sit song write sing change world moment know change world moment know change world moment revolution come minute away people march streets today know hate balance razor blade paint p icture scene know people follow dream reremember life fittin jeans understand pos itivity change world piano bass guitar grab beat away oneman university degree lo rd know everybody talk bout exponential growth stock market crash portfolios sit song write sing change world moment know change world moment know change world mo ment paint picture scene know future hand free spread understand positivity chang e world piano bass guitar grab beat away oneman university degree lord know every body talk bout exponential growth stock market crash portfolios sit song write si ng change world moment know change world moment know change world moment - person al
Index 599: zayn - good years - 2018 - pop - close eye crowd thousand tear pray wa ste good years good years good years voice scream loud hell care bout world bring high star worry want sorry close eye crowd thousand tear pray waste good years go od years good years drug alcohol hell fight cause  world know numb dumb change st ory want sorry close eye crowd thousand tear pray waste good years good years goo d years need breathe feel alive meet night light feel wind hold pain deep inside eye eye close eye crowd thousand tear pray waste good years good years good years pray waste good years good years good years - personal
Index 121: yungblud - hope for the underrated youth - 2019 - rock - leave today t hrow away magazine hair change second juvenile stay heart better sorry better lat e heart better hide feel know dream cause underrate youth tell truth cause pull p ull pull yeah underrate youth episode like fiction mix true lyric commercial - sa dness
Index 285: welshly arms - sanctuary - 2018 - blues - darkness sleep hold close sp ace need spark go glow bring home yeah bring home hurt feel pain dirt wash rain w alk road felt shame place home time changin sanctuary shelter peace sanctuary saf e sanctuary shelter peace sanctuary safe rain start toll slow cause know world tu rn cold need hold hold hold hurt feel pain dirt wash rain walk road felt shame pl ace home time changin sanctuary shelter peace sanctuary safe sanctuary shelter pe ace sanctuary safe yeah yeah safe yeah yeah share hurt share pain dirt wash rain walk road felt shame time changin sanctuary hold sanctuary hold safe sanctuary sh elter peace sanctuary safe sanctuary shelter peace sanctuary safe yeah yeah safe yeah yeah safe sanctuary safe hold cause sanctuary - sadness
Training dataset size: 13 songs
```

In [39]:

```
training_data["document"] = training_data["artist_name"].fillna("") + " " + \
                training_data["track_name"].fillna("") + " " + \
                training_data["genre"].fillna("") + " " + \
                training_data["lyrics"].fillna("")
training_data.reset_index(drop=True, inplace=True)
training_data["clean_lyrics"] = training_data["lyrics"].apply(lambda x: preproce
training_data["clean_text"] = training_data["document"].apply(lambda x: preproce
```

```
week4_df["document"] = week4_df["artist_name"].fillna("") + " " + \
                week4_df["track_name"].fillna("") + " " + \
                week4_df["genre"].fillna("") + " " + \
                week4_df["lyrics"].fillna("")
week4_df.reset_index(drop=True, inplace=True)
week4_df["clean_text"] = week4_df["document"].apply(lambda x: preprocess(x, True
```

Build user profile

In [40]:
```
X_liked_vec = best_vectorizer_tfidf.transform(training_data['clean_text'])
training_data['predicted_topic'] = best_clf.predict(X_liked_vec)

def build_user_profile_from_df(df, vec):
    profile_dict = {}
    for topic in ['dark', 'emotion', 'lifestyle', 'personal', 'sadness']:
        topic_docs = df[df['predicted_topic'] == topic]['clean_text']
        profile_text = ' '.join(topic_docs)
        profile_vec = vec.transform([profile_text])
        profile_dict[topic] = profile_vec
    return profile_dict

user_x_profile = build_user_profile_from_df(training_data, best_vectorizer_tfidf
```

Real user feedback for Week4

In [41]:
```
X_week4_vec = best_vectorizer_tfidf.transform(week4_df['clean_text'])
week4_df['predicted_topic'] = best_clf.predict(X_week4_vec)

sim_scores = compute_similarity_scores(user_x_profile, X_test_vec_tfidf, week4_d

topN_df = get_top_n_recommendations(test_data, sim_scores)

for i, (_, row) in enumerate(topN_df.iterrows(), 1):
    print(f"Index {i}: {row['artist_name']} - {row['track_name']} - {row['releas
    print()

user_real_liked_songs = []

print("\nEnter the index of songs you like (type 'done' to finish):")
while True:
    user_input = input().strip()
    if user_input.lower() == 'done':
        break
    try:
        song_index = int(user_input)
        if song_index in topN_df.index + 1:
            user_real_liked_songs.append(song_index - 1)
            print(f"Selected: Index {song_index}")
        else:
            print("Invalid index, please try again")
    except ValueError:
        print("Please enter a valid number or 'done'")
```

Index 1: justin moore - got it good - 2016 - country - wake morning warn little b ounce blue eye start pillow talk twist sheet finger walk hell start yeah good sur e good see smile kiss eye hips whoah tell like good good sure good good morning g ood late night good thing go good right turn kitchen light cork drink wine right outta bottle babe spin white noise record play little gaye minute bedroom bind ye ah good sure good see smile kiss eye hips whoah tell like good good sure good goo d morning good late night good thing go good right good morning good late night g ood thing go good right yeah good sure good see smile kiss eye hips whoah tell li ke good good sure good good morning good late night good thing go good right - em otion

Index 2: the detroit cobras - shout bamalama - 2016 - blues - alabama shout bamal ama louisiana go lord soul chickens steal night night go unintelligible chicken b aby shout bamalama go feet feet go feild feet step feet heel feet swing knock fee t grin teeth fell tongue stay captain say chicken baby shout bamalama go monkey t ell little gorilla come talk family cry shame mother work chain gang bust bricks unintelligible shout bamalama go preacher deacon pray come come preacher tell dea con prayer say lord prayer want kill baby shout bamalama go - dark

Index 3: thomas rhett - life changes - 2017 - country - wakin college dorm yeah l ife pretty normal lookin date spring formal wasn worry bout nothin majorin undeci ded notebook songs writin dream like sittin walmart shelf funny life change nutti n life change stop train know gonna happen plan hear laughin life change wouldn c hange world world wouldn change world world buy ring say lyric commercial - perso nal

Index 4: timeflies - once in a while - 2016 - pop - think know better wishful thi nk think pressure wishful drink forever feel afraid know brooklyn york cali khale d best strip stay american express live total request question like press road mu sic truth worry bout catch need help feel like wanna dance feel good yeah feel go od feel good good good good yeah feel good live life like blood type positive day s feel opposite music like heroin hear hit time watch somebody copy bind feel fak e like somebody need slow brown thoughts drown think know better wishful think th ink pressure wishful drink forever feel afraid know catch need help feel like wan na dance feel good feel good feel good good good good yeah feel good look go slow think give road slow turn graduate work come raise finish year right move tell to night cause worry catch need help feel like wanna dance feel good yeah feel good feel good good good good yeah feel good wanna dance feel good - emotion

Index 5: taylor swift - i did something bad - 2017 - pop - trust narcissist play like violin look ohsoeasy cause tell tell world work think feel flame skin crimso n paint lips talk regret cause come feel good feel good felt good good dada dada trust playboy world think save comin world work gotta leave leave feel flame skin say throw away good thing drop nothin spend change comin badoh feel good feel goo d felt good good felt good dada dada burn witch aren pitchfork proof receipt reas on burn witch aren light light light light light ahead light light light light li ght light light light light feel good feel good felt good good dada dada feel goo d good feel good feel feel good felt good good - emotion

Index 6: jahmiel - live without limit - 2018 - reggae - yeah yeah yeah yellow moo n clock tick eye drip give time hear life live hard yeah yeah tomorrow promise li ve limit live limit yeah yeah yeah sure bout tomorrow hard nuff know stop yeah af raid live fear greater reason life live hard yeah yeah tomorrow promise live limi t live limit yeah yeah yeah señorita feature yeah like whoa life like speaker thi ng turn load things easier cyaan ease people time hand sit watch people grow time things whoa life live hard yeah yeah tomorrow promise live limit live limit yeah yeah yeah clock tick eye drip give yeah yeah yeah yeah clock tick eye drip give y ellow moon yeah life live hard yeah yeah tomorrow promise live limit live limit y eah yeah yeah - personal

```
Index 7: the band steele - sit awhile - 2017 - country - wake favorite place stud
io hide space cold dark room need feet walk streets headphones beat know go want
yeah yeah shall time lose open road lose lesson learn road choose concern time li
ve time time laugh time time time hate world heavy feel weight know yeah know kno
w yeah know things finally look shoot truth hurt turn world upside catch middle g
o come dream dream high tell live life know certain think agree real life dream y
eah yeah dream dream dream dream - personal

Index 8: daniel caesar - superposition - 2019 - pop - irony things inspire bleed
profusely need time space think breathe mean cash grow tree exist superposition l
ife contradiction yang fluidity things reason sing understand whoa know think  ea
sy ride risk know actin sleazy work easy figure break resuscitate know live life
vain music  piece cake rest life state chaos know okay exist superposition life c
ontradiction yang fluidity things reason sing understand - personal

Index 9: dirty heads - horsefly - 2019 - reggae - say life live know things passi
onate sound real adamant  come sentiment try losses horse race track track offens
e say yeah yeah dream speak cross list feel like feel like thousand reason feel g
ood right good right feel like feel like thousand reason feel good right good rig
ht know cause wanna condolences follow path long path hard apologies need repeat
know cause heart feel revel revel revel second devil devil devil inside head go a
way people style tell real  fake feel like feel like thousand reason feel good ri
ght good right feel like feel like thousand reason feel good right good right fee
l like feel like thousand reason feel good right good right feel like feel like t
housand reason feel good right good right feel like feel like thousand reason fee
l good right good right feel like feel like thousand reason feel good right good
right - emotion

Index 10: chris janson - redneck life - 2017 - country - grow batten board cabin
dead gravel street pay turn cause money grow tree cheap cigarettes windows breath
e choose redneck life redneck life choose yeah daddy build string motor tree wear
beer helmets eye race junkyard dream yeah crew grow eighty choose redneck life re
dneck life choose yeah choose redneck life redneck life choose mountain silver sp
oon week choose redneck life redneck life choose grow swim cutoff jeans bridge ca
stor creek bend half bill bend spend fall sit high tree yeah hunt fish wasn trend
choose redneck life redneck life choose yeah choose redneck life redneck life cho
ose mountain silver spoon week choose redneck life redneck life choose woah - per
sonal


Enter the index of songs you like (type 'done' to finish):
Selected: Index 1
Selected: Index 3
Selected: Index 4
Selected: Index 5
Selected: Index 6
Please enter a valid number or 'done'
Selected: Index 7
Selected: Index 8
Selected: Index 10
```

In [42]:

```python
def evaluate_real_user_feedback(recommendations, liked_indices):
    hits = [1 if i in liked_indices else 0 for i in range(len(recommendations))]
    precision = sum(hits) / len(hits)
    recall = sum(hits) / len(liked_indices) if len(liked_indices) > 0 else 0
    f1 = 2 * (precision * recall) / (precision + recall) if (precision + recall)
    ap = sum([hits[i] * (sum(hits[:i+1]) / (i+1)) for i in range(len(hits))]) /
    return precision, recall, f1, ap

precision, recall, f1_score, map_score = evaluate_real_user_feedback(topN_df, us
```

```
print(f"Precision@N: {precision:.4f}")
print(f"Recall@N: {recall:.4f}")
print(f"F1@N: {f1_score:.4f}")
print(f"MAP@N: {map_score:.4f}")
```

```
Precision@N: 0.8000
Recall@N: 1.0000
F1@N: 0.8889
MAP@N: 0.8228
```

User Feedback and Recommendation Performance Analysis

User Comment

"I like emotional and personal lyrics more. Songs No. 2, 3, and 6 really touched me — they express inner feelings or sufferings of coming of age in life. The ones with more 'negatives' in them, on the other hand, seemed superficial to me, and I bypassed them quickly. Overall, the recommended songs this time were quite close to what I prefer, especially some of the songs in the middle section of the Week 4 list — they hit the spot really well."

Recommendation Metrics

Metric	Value
Precision@N	0.8000
Recall@N	1.0000
F1@N	0.8889
MAP@N	0.8228

The above Recall@N is a hypothetical best case and not really dependable. Since I can only view the user's response from among the Top-N recommended songs, but can't obtain the entire list of songs the user actually liked from all Week 4 test set (250 songs), the recall calculation assumes all the liked songs are within the recommended list. That is too idealized an assumption and fails in real-world cases.

Analysis and Comparison

As opposed to keyword-driven simulated user profiles for Part 2, the user profile developed in Part 3 from real preferences is more precise. The user certainly favored "emotion" and "personal" headed lyrics for the first three weeks, and the recommendation system could focus the Week 4 results on those subjects, indicating good interest modeling. However, obvious constraints remain: with lyrics of same words, the users may not enjoy certain songs due to some differences in rhythm or tone. Therefore, basic TF-IDF similarity cannot always capture genuine preferences.

Summary

The real-user experiment verifies the correctness of the system in topic matching but also indicates its shortcoming in expressing emotions and style diversity. Future work can

incorporate audio features or emotion modeling to enhance recommendation quality.