

Classifying Song Genre from Lyrics

Comparing TF-IDF and MiniLM embeddings

By Hill Zhang

Introduction - Problem and task

Task: predict a song's genre from its lyrics

Dataset

50k English songs with lyrics
subset of MetroLyrics Kaggle dataset

Input

lyrics text for each song

Artist	Song	Genre	Language	Lyrics
12 stones	world so cold	Rock	en	It starts with pain, followed by hate Fueled by the endless questions no one can answer A stain covers your heart and tears you apart Just like a sleeping cancer I don't believe men are born to be killers I don't believe the world can be saved How did you get here and when did it start? An innocent child with a thorn in his heart What kind of world do we live in?

Question

**How far can we get using only lyrics
(no audio)?**

Output

one of 5 genres:

Rock, Pop, Metal, Jazz, Folk

Dataset and cleaning

1. Start from MetroLyrics dataset (~290k songs with lyrics)
2. Drop rows with missing lyrics or missing genre
3. Keep only English songs (Language = "en")
4. Keep the 5 most common genres (Rock, Pop, Metal, Jazz, Folk)
5. Randomly downsample to 50k songs to make training faster

(290183, 5)					
	Artist	Song	Genre	Language	Lyrics
0	12 stones	world so cold	Rock	en	It starts with pain, followed by hate\nFueled ...
1	12 stones	broken	Rock	en	Freedom!\nAlone again again alone\nPatiently w...
2	12 stones	3 leaf loser	Rock	en	Biting the hand that feeds you, lying to the v...
3	12 stones	anthem for the underdog	Rock	en	You say you know just who I am\nBut you can't ...
4	12 stones	adrenaline	Rock	en	My heart is beating faster can't control these...

Train / Val / Test split

- Stratified split by genre
- 60% train (30k), 20% validation (10k), 20% test (10k)
- Same split used for all feature types and models

Split	Size	Purpose
Train	30k	Fit models
Validation	10k	Pick regularization C
Test	10k	Final evaluation (held-out data)

Representation 1

– TF-IDF (bag-of-words)

- Treat each song as a bag of words
- TfidfVectorizer, max_features = 20,000, remove English stop words
- Each song → 20,000-dim sparse vector
- Training matrix shape: 30k songs × 20k features

```
tfidf = TfidfVectorizer(  
    max_features=20000,  
    stop_words="english"  
)  
X_train_tfidf = tfidf.fit_transform(X_train)  
  
X_train_tfidf.shape
```

✓ 1.1s

(30000, 20000)

Representation 2a

– MiniLM text embeddings

- Use a pre-trained sentence transformer (all-MiniLM-L6-v2)
- Input: full lyrics text for each song
- Output: 384-dim dense embedding per song
- Training matrix shape: 30k songs × 384 features

```
# Load a small pre-trained text model, MiniLM
# Input: full lyrics text
# Output: one fixed-length vector per song
embed_model = SentenceTransformer("all-MiniLM-L6-v2")
✓ 11.8s

# Compute embeddings for train, validation, and test sets

X_train_embed = embed_text_list(X_train)
X_val_embed = embed_text_list(X_val)
X_test_embed = embed_text_list(X_test)

X_train_embed.shape
✓ 4m 14.4s

(30000, 384)
```

Representation 2b

– MiniLM + autoencoder (64-dim latent)

- Start from the 384-dim MiniLM lyric embeddings.
- Train a small autoencoder to reconstruct them:
 $384 \rightarrow 128 \rightarrow 64 \text{ (latent)} \rightarrow 128 \rightarrow 384$.
- Use the 64-dim latent vector as another feature set (“MiniLM-AE”) for genre prediction.
- Question: Does this 64-dim compressed representation beat the raw MiniLM features?

Layer	Input dim	Output dim
Encoder Linear 1	384	128
Encoder Linear 2	128	64 (<i>latent</i>)
Decoder Linear 1	64	128
Decoder Linear 2	128	384

Model and training setup

- For each feature set (TF-IDF, MiniLM, MiniLM-AE), train one multinomial logistic regression model for 5 genres
- Linear model, works well in these high-dimensional feature spaces
- Tune regularization $C \in \{0.1, 1, 10\}$ on the validation set for each feature type
- Pick the C with the best validation macro-F1, retrain on train+val, then report test accuracy and macro-F1

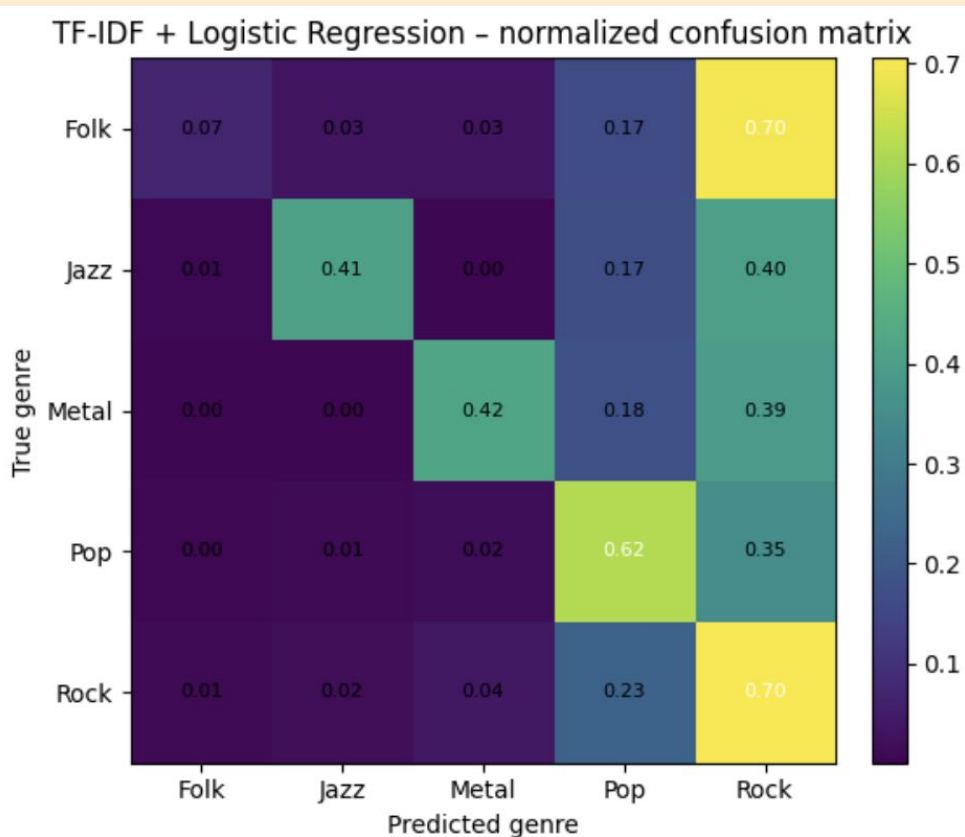
Results – logistic regression on three feature sets

- Same train/val/test split and logistic model for all three feature sets
- TF-IDF (20k features) gets the best macro-F1, at about 0.61 accuracy
- MiniLM (384-dim) has similar accuracy but lower macro-F1
- MiniLM-AE (64-dim latent) drops a bit more → compression throws away some genre information

Features	Test acc	Test macro-F1
TF-IDF (20k)	0.6096	0.4732
MiniLM (384)	0.6090	0.4426
MiniLM-AE (64 dim)	0.5939	0.3938

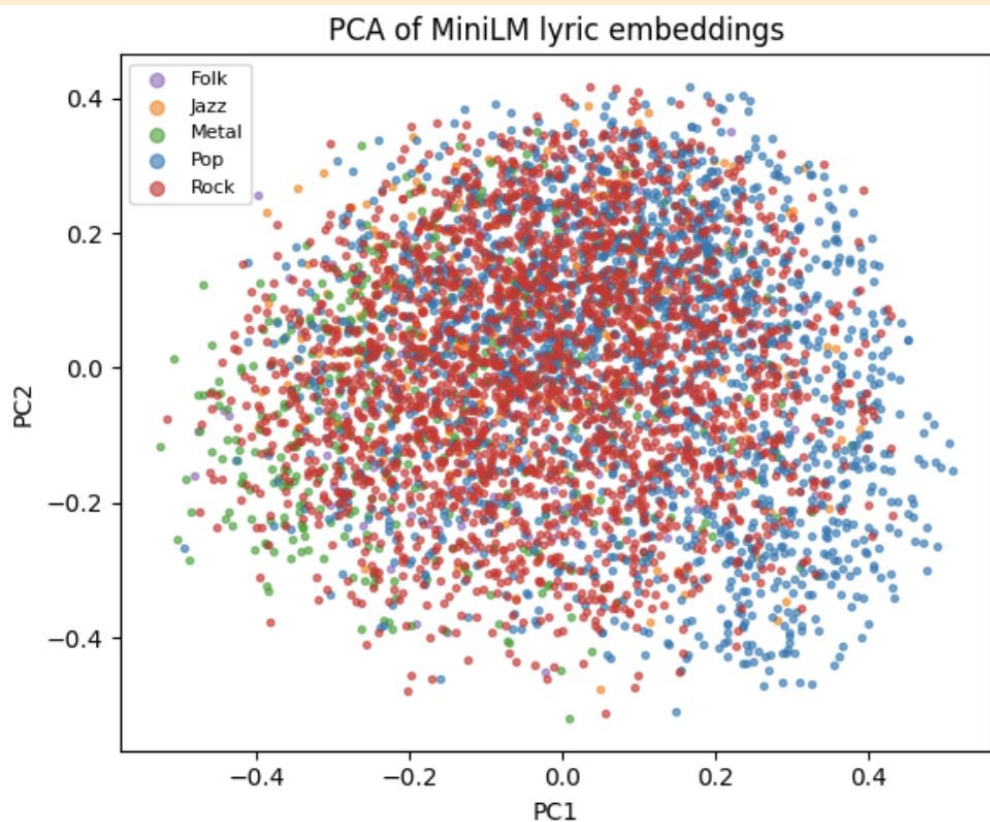
TF-IDF + Logistic Regression – Confusion Matrix

- Using the best model (TF-IDF + logistic regression)
- Rows are true genres, columns are predictions; each row is normalized to 1
- Folk is tiny and often gets pushed into Rock → explains low macro-F1
- Pop and Rock are mostly correct; Metal and Jazz are in between



PCA of MiniLM embeddings

- PC1 vs PC2 of the 384-dim MiniLM lyric embeddings, colored by genre
- Rock and Pop overlap a lot → lyrics do not cleanly separate genres
- Smaller genres (Jazz, Metal, Folk) show up as small clouds mixed into the big cluster
- This helps explain why all models sit around ~0.6 accuracy



What I learned

- TF-IDF + logistic regression is still very strong and slightly outperforms both MiniLM and MiniLM-AE on macro-F1.
- High dimensional linear models work in practice: a $30k \times 20k$ sparse TF-IDF matrix trained smoothly.
- MiniLM + autoencoder gives a compact 64 dimensional latent space, but it does not beat simple TF-IDF on this task.
- Newer models are not automatically better; strong baselines still matter and need to be tested.

THANK YOU