Lyircs-based Author Classification set up

programs/create_artist_dataset.ipynb **Outputted files** lyrics/AAMDS/*artistname*.csv --> contains eight Uses the Genius API to create the data set of a Afro-american male artists specific artist. Ivrics/CADSa/*artistname*.csv --> contains two Filtering is applied to filter out unintended instances Afro-american male artists and their collaborative such as interviews, album art explanations etc. Writes to a csv file with columns: lyrics/CADSb/*artistname*.csv --> contains an lyrics Afro-american and Jamaican artist and their - song title collaborative album - artist name - featuring artists lyrics/DADS/*artistname*.csv --> contains four afro-american male artists, two white american male artists and two Afro-american female artists lvrics/GBDS/*gender*/*artistname*.csv --> contains seven female and seven male artists programs/preprocess_lyrics.py **Outputted files**

Imports the outputted files from create_artist_dataset.ipynb and creates a train, dev and test set datasets/AAMDS_*trainIdevItest*.csv datasets/CADSa_*trainItest*.csv --> due to the nature of the set their is no development data, is

Distinguishes between full song lyrics and verse lyrics

Applies tests to test validity of verses

explained in the paper

datasets/CADSb_*trainltest*.csv --> due to the nature of the set their is no development data is

Applies tests to test validity of verses

nature of the set their is no development data, is explained in the paper

Preprocesses lyrics to remove all irrelevant characters, such as comments, some punctuations

datasets/DADS *trainidevitest* csv.

characters, such as comments, some punctuations etc.

datasets/DADS_*trainIdevItest*.csv

datasets/GBDS_*trainIdevItest*.csv

Writes these to csv files.

programs/optimize_each_feat.py

Determines the optimal settings for each feature through manual grid searches

For each feature:

of each lyrics

- check word n-grams up to 3-6
- check char and char_wb n-grams up to 9-13
- check for min document frequences of 5 and 10
- check for count or tfidf vectorizer

Save the macro-f1 score based on kvold(9) for each possible combination in a dictionairy and save the

Outputted files

datasets/settings/AAMDS_tried_settings.p datasets/settings/CADSa_tried_settings.p datasets/settings/CADSb_tried_settings.p

datasets/settings/DADS_tried_settings.p

datasets/settings/GBDS_tried_settings.p

highest word and char(_wb) settings per feature while running.	
programs/create_best_feat_set.py Determines the best combination of features by: - Starting with the highest macro-f1 scoring feature - Add the second best feature to the active feature set - Classify with kvold = 9 - If the macro-f1 score improves add the feature to the feature set Add the end the best feature set is saved to a file	Outputted files datasets/settings/AAMDS_best_feat_set.p datasets/settings/CADSa_best_feat_set.p datasets/settings/CADSb_best_feat_set.p datasets/settings/DADS_best_feat_set.p datasets/settings/GBDS_best_feat_set.p
programs/classify.py Loads the best settings for a dataset Classifies using these settings on the dev, test or kvold on the training set and outputs the result	Outputted files