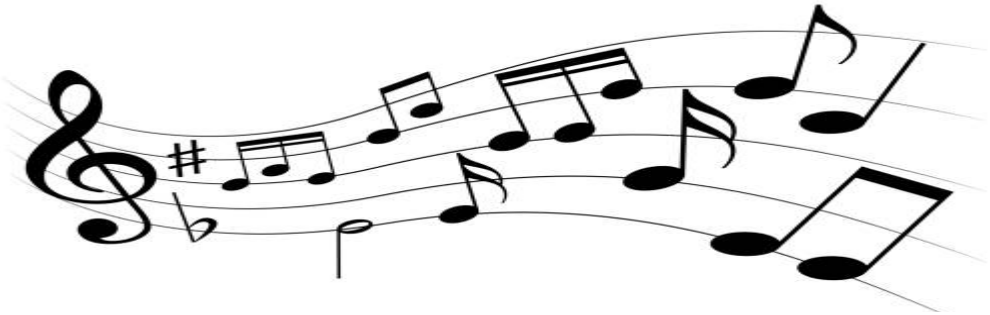
Text Classification

Guess Artist from Lyrics



1- Get Data: LyricsGenius

- LyricsGenius: a Python client for the Genius.com API

- lyricsgenius provides a simple interface to the song, artist, and lyrics data stored on Genius.com.

- **Installation:** pip install lyricsgenius

- **Usage:** Import the package and initiate Genius:

```
import lyricsgenius
genius = lyricsgenius.Genius(token)
```

API Client

genius.com/api-clients#

Documentation

Support Forum

TOS

All API Clients

New API Client

API Clients

Lyrics-Artist Edit

APP WEBSITE URL

https://github.com/spicedacademy/tensor-tarragon-student-code/tree/Mahmoud

CLIENT ID

n99AGo0JB4cpSRjAIh15F1AQbERD9YSMN5DureP25eSsp6UJSay6N3j

CLIENT SECRET

This key is secret! Hover to view and copy

CLIENT ACCESS TOKEN

olA4K9BY5VHZEnJTHOlGmA_uWwcbN71sob1h9MzcVmWhzPT_laG

Collect songs

```python
#define list of artists
artists = ['Frank Sinatra','Ed Sheeran','Taylor Swift']
max_songs=150



#search for songs for each artist in the list and save the songs to the artist folder
def collect_songs(artists,max_Songs):

    api=genius.Genius('r2JOC0T6SWh2hLSP5yWaTu2f5LkXbJXc-mMvNg1rE7VDYGoBt-jNcFxRnYbKUjGS',
                      excluded_terms = ["(Remix)", "(Live)"] ,
                      skip_non_songs=True,
                      remove_section_headers=True)

    for artist in artists:
        songs = (api.search_artist(artist, max_songs=max_Songs, sort='popularity')).songs
        c=0
        for song in songs:
            fileName = os.path.join('lyrics/'+artist,"songnumber"+str(c)+".txt")
            file = open(fileName, "w")
            file.write(song.lyrics)
            file.close()
            c+=1
```

Collect Songs

```
In [9]: collect_songs(artists,max_songs)
```

```
Searching for songs by Frank Sinatra...

Song 1: "Fly Me to the Moon"
Song 2: "My Way"
Song 3: "That's Life"
Song 4: "New York, New York"
Song 5: "Somethin' Stupid"
Song 6: "The Girl From Ipanema (Garota De Ipanema)"
Song 7: "The Way You Look Tonight"
Song 8: "Come Fly with Me"
Song 9: "Have Yourself a Merry Little Christmas"
Song 10: "I've Got You Under My Skin"
Song 11: "My Funny Valentine"
Song 12: "It Was a Very Good Year"
Song 13: "Strangers in the Night"
Song 14: "Blue Moon"
Song 15: "Luck Be a Lady"
Song 16: "You Make Me Feel So Young"
Song 17: "Let It Snow! Let It Snow! Let It Snow!"
```

Correct Artist name

```
In [4]: collect_songs(artists,max_songs)
```

```
Searching for songs by ed sheran...

Changing artist name to 'Ed Sheeran'
Song 1: "Shape of You"
Song 2: "Perfect"
Song 3: "Castle on the Hill"
```

2- Data Cleaning

- 1- Natural language toolkit (NLTK)
- 2- SpaCy

- NLTK is essentially a string processing library, where each function takes strings as input and returns a processed string.But spaCy takes an object-oriented approach.

- **Which is better?**

spaCy provides the best way to do it. It provides the fastest and most accurate syntactic analysis of any NLP library released to date.

Data Cleaning with SpaCy

```python
# Apply spacy to the text
doc=nlp(text)
# Lemmatization,remove noise (stopwords, digit, puntuaction and single characters)
tokens=[token.lemma_.strip() for token in doc if
    not token.is_stop and not nlp.vocab[token.lemma_].is_stop # remove StopWords
    and not token.is_punct # Remove puntuaction
    and not token.is_digit # Remove digit
    and not token.is_space
    and not token.is_quote
    and not token.is_bracket
    and not token.like_num
    and not token.is_currency
   ]
# Remove empty tokens and one letter tokens
tokens = [token for token in tokens if token != "" and len(token)>1]
# Recreation of the text
new_text=" ".join(tokens)
# Remove non alphabetic characters
new_text = re.sub(r"[^a-zA-Z]", " ", new_text)
# remove non-Unicode characters
new_text = re.sub(r"[^\x00-\x7F]+", "", new_text)

new_text=new_text.lower()
```

Data Cleaning with SpaCy

songnumber36.txt

```
Things were all good yesterday
And then the devil took your memory
And if you fell to your death today
I hope that heaven is your resting place
I heard the doctors put your chest in pain
But then that could've been the medicine
And now you're lying in the bed again
Either way I'll cry with the rest of them

And my father told me "son
It's not his fault he doesn't know your face
And you're not the only one"
Although my grandma used to say that he used to sing

Darling, hold me in your arms the way you did last night
And we'll lie inside for a little while here, oh
I could look into your eyes until the sun comes up
And we're wrapped in light and life and love
Put your open lips on mine and slowly let them shut
For they're designed to be together, oh
With your body next to mine, our hearts will beat as one
And we're set alight, we're afire love
Oh love
```

song_cleaned_number36.txt

```
thing good yesterday devil memory fall death today hope heaven rest place hear doctor chest pain medicine lie bed way cry rest father tell son fault know face grandma sing darling hold arm way night lie inside little oh look eye sun come wrap light life love open lip slowly let shut design oh body heart beat set alight afire love oh love thing good yesterday devil breath away leave pain black suit black tie stand rain family staple stranger friend come mind paint pen year old remember father tell son fault know face grandma sing darling hold arm way night lie inside little oh look eye sun come wrap light life love open lip slowly let shut design oh body heart beat set alight afire love oh love father family rise seat sing hallelujah brother family rise seat sing hallelujah brother family rise seat sing hallelujah brother sister father family rise seat sing hallelujah
```

Train a Model

- Create CORPUS

- Vectorize using TfidfVectorizer

- train with:

❖ - Random Forest Classifier

❖ - Multinomial Naive Bayes

- Use GridSearch for Hyperparameters tuning

Train with Random Forest Classifier

```python

"""train model with Random Forest classifier + GridSearch"""
def train_model_RF_GridSearch(X_train,y_train):
    pipeline = Pipeline([
        ('Tfidf',TfidfVectorizer()),
        ('RF',RandomForestClassifier())
    ])

    params = {
    'Tfidf__max_features':[1000,2000],
    'Tfidf__ngram_range': [(1, 1), (1, 2), (1, 3)],
    'Tfidf__min_df':[5,7,10],
    'Tfidf__max_df':[0.5,0.6,0.7],
    'RF__n_estimators':[1000,2000],
    'RF__max_depth':[3,5,7],
    }

    tfidf_gs = GridSearchCV(pipeline, param_grid=params, cv = 5, verbose = 1,scoring='accuracy',n_jobs=-1)
    tfidf_gs.fit(X_train,y_train)
    print('Best parameters:',tfidf_gs.best_params_)
    best_model=tfidf_gs.best_estimator_
    return best_model
```

Evaluate RFC

```
Model accuracy: 0.81


confusion matrix
[[19  3  1]
 [ 2 33  0]
 [ 8  3 21]]


classification report
              precision    recall  f1-score   support

   Ed Sheeran       0.66      0.83      0.73        23
Frank Sinatra       0.85      0.94      0.89        35
 Taylor Swift       0.95      0.66      0.78        32

     accuracy                           0.81        90
    macro avg       0.82      0.81      0.80        90
 weighted avg       0.84      0.81      0.81        90
```

Train with MultinomialNB

```python

""" train model with Naive bias using GridSearch"""
def train_model_NB_GridSearch(X_train,y_train):

    pipeline = Pipeline([

    ('Tfidf',TfidfVectorizer(stop_words='english')),
    ('NB',MultinomialNB())
    ])
    params = {
    'Tfidf__max_features':[1000,2000,4000],
    'Tfidf__ngram_range': [(1, 1), (1, 2), (1, 3)],
    'Tfidf__min_df':[5,7,10],
    'Tfidf__max_df':[0.5,0.6,0.7],
    'NB__alpha':[1,0.5,0.1, 0.01, 0.001, 0.0001],
    }

    tfidf_gs = GridSearchCV(pipeline, param_grid=params, cv = 5, verbose = 1,scoring='accuracy',n_jobs=-1)
    print('start training\n')
    tfidf_gs.fit(X_train,y_train)
    print('Best parameters:',tfidf_gs.best_params_)
    best_model=tfidf_gs.best_estimator_
    return best_model

```

Evaluate MultinomialNB

```
start training

Fitting 5 folds for each of 486 candidates, totalling 2430 fits
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 4 concurrent workers.
[Parallel(n_jobs=-1)]: Done  42 tasks      | elapsed:   10.9s
[Parallel(n_jobs=-1)]: Done 192 tasks      | elapsed:   29.2s
[Parallel(n_jobs=-1)]: Done 442 tasks      | elapsed:   58.0s
[Parallel(n_jobs=-1)]: Done 792 tasks      | elapsed:  1.6min
[Parallel(n_jobs=-1)]: Done 1242 tasks      | elapsed:  2.5min
[Parallel(n_jobs=-1)]: Done 1792 tasks      | elapsed:  3.6min
[Parallel(n_jobs=-1)]: Done 2430 out of 2430 | elapsed:  4.8min finished
Best parameters: {'NB__alpha': 0.1, 'Tfidf__max_df': 0.5, 'Tfidf__max_features': 2000, 'Tfidf__min_df': 5, 'Tfidf__ngram_range': (1, 2)}

 Model accuracy: 0.72
```